SOME DECENTRALIZED OPTIMIZATION AND CONTROL ALGORITHMS FOR THE CONTROL OF WIND FARMS

BY

ALBERT Z. CHENG

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Mechanical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2011

Urbana, Illinois

Adviser:

Professor Cédric Langbort

Abstract

This is a preliminary study of decentralized algorithms that can be applied to wind farm controls. Traditionally, wind farm control is comprised of the wind farm level control and the wind turbine level control. The wind farm level control is a centralized controller that takes the demands from the grid and generates operating points for each wind turbine within the wind farm. The wind turbine level control then generates the optimal control for each turbine to match the operating point. Unfortunately, this traditional control scheme does not constitute the optimal operation of a wind farm due to it's disregard at either level of control for the interactions between wind turbines in the wind farm .

Consequently, a different two level control scheme is proposed in this thesis. This control scheme is shown to be a decentralized controller in that each wind turbine has the ability to both generate its own operating point and calculate its own optimal control. Through the communication of the wind turbines with each other, the interactions between the wind turbines are incorporated into both levels of control. The generation of the operating point is posed as a stochastic resource allocation problem that takes into account the stochastic wind and other wind farm characteristics. We develop a stochastic algorithm based on network dynamic system theory to solve the resource allocation problem. We show that the algorithm converges to the solution of the resource allocation problem almost surely. The calculation of each wind turbine's optimal control is formulated as an Linear Quadratic Regulator (LQR) optimization problem with a equality constraint. We develop an algorithm that is based on the Tatonnement process in Economics to solve the LQR problem. We first consider the performance of the algorithm in a dynamically decoupled system and show that the algorithm solves the LQR problem. We then consider the performance of the algorithm in a dynamically coupled system and discuss the difference between the two cases.

To Father and Mother.

Acknowledgments

I would like to thank to my adviser, Professor Cédric Langbort, for his patience and his guidance throughout my Master program. He helped me immensely in understanding and appreciating many of the ideas in control theory. I also like to thank my group mates Takashi Tanaka, Abhishek Gupta, Heather Arneson, and Andres Ortiz for their support and help through these past two years. Furthermore, I would like to thank my math teachers throughout the years in preparing me with the knowledge to do this work. Finally, I would like to thank my parents for their continuing love and support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Despite recent economic hardships, the global wind energy market has been booming. According to [8], the annual global wind energy market grew 41.5% from 2008-2009. In 2009, the global wind turbine installation market was worth 63 billion US dollars. The United States is a global leader in installation capacity, increasing the country's installation capacity by 39% in 2009. Currently in the US, wind energy is generating close to 2% of US electricity, but the US Department of Energy in 2008 reported that wind power has the potential to provide as much as 20% of US electricity by 2030.

Because of the increasing number of wind turbine installation, a majority of the wind turbines in the world is now arranged in wind farms or arrays. An advantage of wind farms is that they reduce the average cost of energy due to economies of scale [14]. As the number of wind farms continue to increase, it is important to consider the optimal operation of wind farms that maximizes the efficiency in energy production. The efficiency of the wind farm is defined as

$$\eta_A = \frac{E_A}{E_T N}, \tag{1.1}$$

where η_A is the efficiency of the wind farm A, E_A is the annual energy produced by the wind farm A, E_T is the annual energy produced by a single isolated wind turbine in the wind farm, and N is the number of wind turbines in the wind farm [25]. Therefore, we need to maximize the energy generation of the entire wind farm in order to maximize the efficiency of the wind farm.

1.1.1 Wind Turbine Model

In discussing the optimal operation of the wind farm, we need to first understand the optimal control of a single wind turbine. We limit our discussion to the most common type of wind turbines, namely the variable speed horizontal-axis wind turbines (HAWTs). Furthermore, although modern turbine control includes the control of pitch angle of the turbine blade to minimize aerodynamic stress, we neglect the discussion here since our interest is the maximization of energy generated by the turbine.

There are four operational regions that are dependent on the wind speed for a variable speed turbine. Region 1 is characterized by the region from zero wind speed to

the cut in speed. The cut in speed is the wind speed at which the power within the wind exceeds the power loss by the wind turbine system, and therefore it is economical to operate the wind turbine. Region 2 is the wind turbine operation region for wind speeds between the cut in speed and high wind speeds. At Region 3, high wind speeds cause the wind turbine to limit its energy generation so as to avoid exceeding electrical and mechanical load limits. These three regions are shown in Figure 1.1. There is also a 4th region, not shown in Figure 1.1, that begins with the cut out speed, which is the speed at which operation of the wind turbine is too dangerous and therefore is stopped.

For wind turbine control, we are most interested in Region 2. In this region, we control the rotational speed of the turbine to maximize the turbine's aerodynamic efficiency. The aerodynamic efficiency, also called the power coefficient, of the wind turbine is defined as

$$C_p = \frac{P}{P_{wind}},$$

where P is the power captured by the wind turbine, and P_{wind} is the potential power available in the wind. Furthermore, P_{wind} is defined as

$$P_{wind} = \frac{1}{2}\rho A v^3,$$

where ρ denotes the air density, A is the swept area of the rotor, and v is the wind speed. The swept area of the rotor is the area of the circle with the radius equal to the rotor radius R. We assume that the wind speed is uniform across the rotor swept area. For a modern HAWT, the power coefficient C_p is a function of the tip speed ratio λ, which is defined as

$$\lambda = \frac{\omega R}{v},$$

where ω is the rotational speed of the rotor. The power coefficient of C_p is a nonlinear function of λ that is turbine specific. Therefore, the power captured by the wind turbine can be defined as a function of λ such that

$$P(\lambda) = \frac{1}{2}\rho A v^3 C_p(\lambda). \tag{1.2}$$

It follows that to maximize the generated energy, the control objective of a variable-speed wind turbine controller is to find an optimal λ^* which maximizes C_p for a wind speed v and drive λ to λ^* through controlling ω. There are many generator torque controllers in industry, and [25] shows a simple torque controller used at National Renewable Energy Laboratory.

1.1.2 Wind Farm Control

Traditionally, wind farms have been controlled by a combination of a centralized control on the wind farm level and individual control on the wind turbine level [12, 17].

The wind farm level controller receives the demands of the consumers through the system operator. The wind farm level controller also acquires wind resource information from measurements of wind profile of the entire wind farm. Using these information, the wind farm controller generates an appropriate power reference and assign it to each wind turbine in the wind farm. Each wind turbine then uses its individual wind turbine level controller to maximize its energy production based on the power reference. This type of control strategy assumes that the maximization of energy generated by each wind turbine in a wind farm results in the maximum efficiency for that wind farm. The traditional control scheme is shown in Figure 1.2.

However, studies have shown that such a control strategy is not the optimal strategy. The example in [32] suggests the operation of a wind farm with three rows of wind turbines by operating the first row of turbines with a λ that is a factor below the optimal λ^*, operating the second row of turbines with a even smaller λ, and operating the last row of turbines with their optimal λ^*. Although the selection of λ's are determined by trial and error, and the authors in [32] do not detail their specific approach, they show that the result is more efficient than operating all turbines with the the optimal tip speed ratio λ^*. The result in [32] indicates that optimizing the energy generation of each individual wind turbines does not necessarily leads to the optimization of the wind farm's energy generation.

One explanation for this difference can be attributed to aerodynamic wake interactions between wind turbines. The spacing between the wind turbines causes the aerodynamic wakes produced by each turbine to interact with other turbines. Wind turbines produce energy by extracting the kinetic energy from the wind. Aerodynamic wakes decrease the wind speed behind each turbine. Therefore, the turbines that interact with these wakes extracts energy from a slower wind speed, and consequently the power productions from these turbines are lowered. In fact, depending on the arrangement of a wind farm, the overall loss of the wind farm can be in the order of 5 to 10% [31].

A simple solution in an attempt to eliminate the effects of wakes is through the spacing of the turbines in each wind farm. Wind turbines are typically spaced farther apart in downwind spacing than in crosswind spacing. Downwind spacing is the spacing of the wind turbine in the direction parallel to the prevailing wind direction, and crosswind spacing the spacing in the direction perpendicular to the prevailing wind direction. Usually, downwind spacing is about 8 to 10 rotor diameters, compared to 4 to 5 rotor diameters for crosswind spacing. All spacings of wind turbines are dependent on the geography where the wind farm is located. However, wind turbines can slow wind up to a distance 5-20 km. Therefore, it is impossible to completely eliminate wake interaction.

More recently, the Energy Research Center in Netherlands (ECN) characterized the aerodynamic wake interaction between wind turbines through the effects of Heat and Flux [7]. At the end of the year 2000, through a derivation from the theoretical situation where the losses due to "technical imperfections" are disregarded, ECN found that actuator discs, as defined by the standard actuator disc model (for example, the model used in [32]) for a wind turbine, extract 50 % more energy from wind than what they

extract as useful power. This is different from the classical thought that useful power is about the same as extracted power. ECN attributes the 50 % power loss to heat dissipation in the wake behind the disc. In further analysis of heat phenomenon in 2002, ECN ran a simulation where the heat effect was switched off. They observed that turbines still produced more power (less than with heat effect switched on) when windward side turbines are operated at less than optimal. ECN characterized this effect as Flux. Through the use of scaled-down custom model turbines in a wind tunnel, the authors in [7] predicts that wind farm productions can increase by 2%.

From these studies, it has become apparent that to maximize energy generation and consequently wind farm efficiency, we need to develop a coordinated control scheme for all wind turbines in a wind farm. Although ECN has made some headway in understanding the physical interactions between wind turbines in a wind farm, such interactions are still not well-understood. Consequently, there is a need for the coordinate control of turbines to be flexible and adaptive to the individual's environment. Fortunately, the structure of a wind farm is relatively decentralized, and therefore a decentralized optimized control strategy seems appropriate. Decentralized control gives wind turbines the ability to work together through communication, thereby allowing decentralize implementation of strategies that optimize the entire wind farm, such as strategies that can exploit the wake interactions between the turbines. A decentralized control scheme can also provide more flexibility for wind turbines, since each individual wind turbine makes its own control decision. This flexibility is important since wind is a deterministically unpredictable energy source. Finally, and perhaps most importantly, practice in industry relies on distributed control structures, and consequently there is a need to develop systematic tools for designing such distributed controllers [17].

1.2 Purpose and Overview of Thesis

In this thesis, we assume that the control objective of a wind farm is to maximize its energy production by minimizing the cost of energy generation. A particularly important constraint of each controlling scheme is the constraint of demand matching, which is defined as the matching of the energy produced to the demand. Therefore,we require an equality constraint associated with the optimization for each of the controllers that we investigate. The purpose of this thesis is to suggest two distributed algorithms that can be developed into a two level distributed wind farm controller. The first level deals with finding the optimal operating point of the wind turbine taking into account the characteristics of the entire wind farm, such as dynamic couplings of aerodynamic wakes. The second level deals with optimizing the wind generation of the wind turbines in the wind farm. Although the proposed algorithms currently cannot be directly applied to wind farms, they serve as a basis for new wind farm controller development with future research.

The rest of the thesis is organized as follows. Chapter 2 formulates the stochastic and deterministic optimization problems that will be solved at each level of the control scheme to minimize the cost of energy generation in a wind farm. Chapter 3 gives a

distributed stochastic control algorithm that solves the stochastic optimization problem. This algorithm can generally be applied to any network systems, and we will discuss how it can also be applied to a wind farm. Chapter 4 provides a Tatonnement-based algorithm that solves the deterministic optimization problem and shows its convergence. We also introduce a non-Tatonnement process as inspiration for future development of control algorithms. Finally, we discuss future challenges these two control algorithms needs to solve to be applicable in wind farm control in Chapter 5.

1.3 Figures

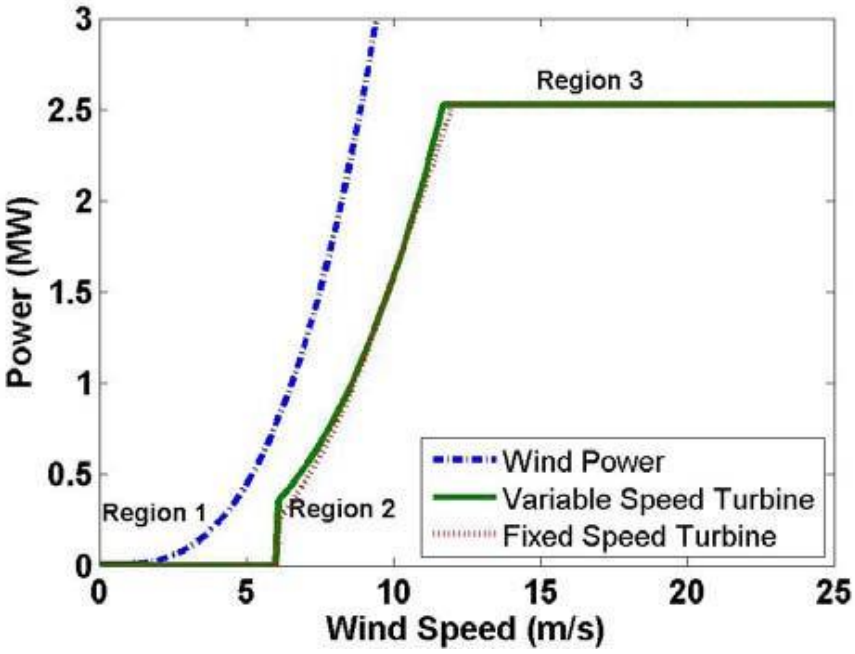

Figure 1.1: Typical power regions for the control of wind turbines, taken from [25].

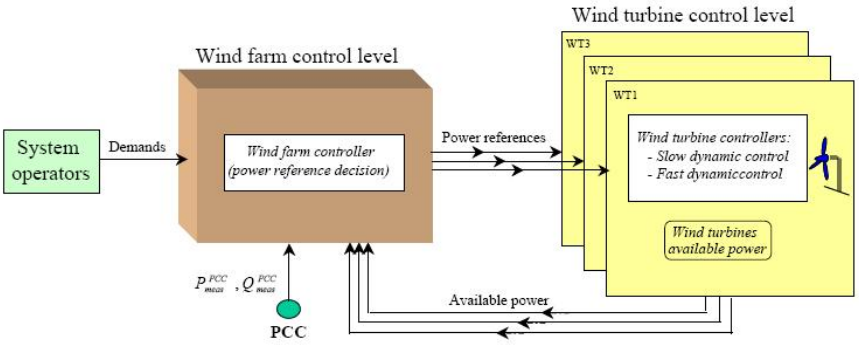

Figure 1.2: The organization of the traditional way to control wind farms, taken from [12].

Chapter 2

Problem Formulation for a Wind Farm Control Scheme

2.1 Introduction

Our control objective is to minimize the cost of energy generation of a wind farm while guaranteeing that the generated energy equals to demand. In the traditional wind farm control, the wind farm controller generates the operating points for each wind turbine, and each wind turbine optimizes itself to generate enough energy to meet these operating points. We propose a two level control scheme where a wind turbine in the wind farm both generates its operating points and optimizes itself to generate enough energy for these operating points through a two level control scheme. This scheme is shown graphically in Figure 2.1

In this chapter, we present the precise mathematical problem formulations for the two levels of controls. This will set the stage for the discussion of our control algorithms that solve these two problems. In the first level, we model the calculation of the operating points as a stochastic convex optimization problem that takes into account the stochasticity of wind and wind farm characteristics. By wind farm characteristics, we mean the properties that form as a result of the arrangement of wind turbines into a wind farm, such as the Heat and Flux phenomenon. Note that this is different from the generation of the typical operating points in [12] since the operating points traditionally is generated without the consideration of wind farm characteristics. This optimization problem is subject to an equality constraint to match energy demand from the network. In the second level, we model the wind farm cost of energy generation function by summing the cost of energy generation functions of each individual wind turbine in the wind farm. We want to find the controller for each wind turbine that optimizes this aggregated cost function subject to an equality constraint that again represent the satisfaction of the energy demand by the generated energy. In this preliminary study, this problem is formulated as a Linear Quadratic Regulator (LQR) optimization problem.

2.2 Stochastic Problem Formulation

In the first level of our control scheme, we are essentially trying to decide the optimal amount of energy each wind turbine should produce at every time step given the wind farm characteristics, the available wind, and the demand from the grid. Because our knowledge of the amount of energy each wind turbine can generate at a specific time

becomes uncertain due to the stochastic nature of wind, our minimization problem at each time step is formulated as a stochastic resource allocation problem with an equality constraint if we consider energy generation as a stochastic resource. Therefore, assuming that the wind turbine system can optimize its own energy generation given an operating point, we formulate a static problem which aims to generate appropriate operating points for each wind turbine.

The stochastic resource allocation problem has a broader application in network system problems. Therefore, we will formulate the problem in terms of a network system problem. Let us consider a network of autonomous agents modeled as a strongly connected directed graph (V,E), where $V = \{1...n\}$ is the set of nodes, and $E \subseteq V \times V$ denotes the edge set. Each edge (i,j) is an ordered pair of distinct nodes. We define $N_i = \{j|(i,j) \in E\}$ as the set of out neighbors of node i. This network model constrains the way in which agents share information with each other, namely i can pass information to j only if $(i,j) \in E$.

In addition, we assume that each agent in the network is subjected to noises from computation and communication with other agents. To characterize these noises, let us define a probability space $(\Omega, \mathscr{F}, P)$, where Ω is the set of outcomes, $\mathscr{F}$ is the σ-algebra generated by the open sets of Ω, and P is the probability measure. Let $(\mathscr{F}(k))_{k\in\mathbb{N}}$ be an increasing family of sub-σ-algebras of $\mathscr{F}$ such that $\cup_{k\in\mathbb{N}}\mathscr{F}(k) = \mathscr{F}$.

We model the computation and communication noises as two real valued random processes

$$\begin{aligned} v_{m_i} &: \Omega \times \mathbb{N} \to \mathbb{R} \quad \forall i \in \{1,\cdots,n\} \\ v_{c_{j\to i}} &: \Omega \times \mathbb{N} \to \mathbb{R} \quad \forall i,j \in \{1,\cdots,n\}, \quad i \neq j. \end{aligned}$$

For simplicity, we define the random variables $\{v_{m_i}(k)\}$ and $\{v_{c_{j\to i}}(k)\}$ as

$$\begin{aligned} [v_{c_{j\to i}}(k)]_{(\omega)} &= v_{c_{j\to i}}(\omega,k). \quad \forall \omega \in \Omega \\ [v_{m_i}(k)]_{(\omega)} &= v_{m_i}(\omega,k) \quad \forall \omega \in \Omega \end{aligned}$$

for all k. These random variables are measurable with respect to $\mathscr{F}(k)$ for all k.

The noises characterize by the random processes$(v_{m_i}(k))_{k\in\mathbb{N}}$ and $(v_{c_{j\to i}}(k))_{k\in\mathbb{N}}$ have properties that result from our network model and our application. Each agent's computational ability, and consequently computation noise, is a unique property of that specific agent because we are assuming that each agent is autonomous. Also, the noise associated with each communication channel is independent of all other channel in the network. Finally, the cause of computation noise is generally different from the cause of communication noise in application. We summarize these properties and make some standard assumptions in Assumption 1.

Assumption 1. *We assume that*

1. *the random processes $(v_{m_i}(k))_{k\in\mathbb{N}}$ and $(v_{c_{j\to i}}(k))_{k\in\mathbb{N}}$ are both i.i.d.*
2. *$\mathbb{E}\{v_{m_i}(k)\} = 0$ and $\mathbb{E}\{v_{c_{j\to i}}(k)\} = 0$ for all i,j,k and $i \neq j$.*

3. $v_{m_i}(k)$ *and* $v_{m_j}(l)$ *are independent* $\forall i \neq j, \forall k, l$.

4. $v_{c_{j\to i}}(k)$ *and* $v_{c_{p\to o}}(l)$ *are independent* $\forall (i,j) \neq (o,p), \forall k, l$.

5. $v_{c_{j\to i}}(k)$ *and* $v_{m_i}(l)$ *are independent* $\forall i \neq j, \forall k, l$.

A real variable $x_i \in \mathbb{R}$ and a convex objective function f_i are associated with each node i in the network. Our goal is to solve the following problem taking into account the computation and communication noises of each agent:

$$\min_{x_1,\ldots,x_n} \sum_{i=1}^{n} f_i(x_i) \quad s.t. \quad \sum_{i=1}^{n} x_i = c,$$

for some constant $c \in R$. For simplicity of notation, we define

$$x = [x_1, x_2, ..., x_n]', \quad F(x) = \sum_{i=1}^{n} f_i(x_i), \quad \hat{1} = [1, 1, ..., 1]'$$

with $'$ denoting the transpose of a vector. Our minimization problem then becomes

$$\min_x F(x) \quad s.t. \quad \hat{1}'x = c. \tag{2.1}$$

We make the following assumptions about the objective function of each agent.

Assumption 2. *We assume that*

1. $f_i : \mathbb{R} \to \mathbb{R}^+$ *is a twice continuously differentiable non-negative function.*

2. *there exists a real number* $u_i \in \mathbb{R}^+$ *such that* $\frac{d^2}{dx_i^2} f_i(x_i) \leq u_i$ *for all* x_i.

3. f_i *is a strictly convex function, i.e.,*

$$f_i(\alpha x + (1-\alpha)y) < \alpha f_i(x) + (1-\alpha) f_i(y)$$

for all x, y *and* $\alpha \in (0,1)$.

4. f_i *is coercive for each i, i.e.,*

$$\lim_{|x_i|\to+\infty} f_i(x_i) = +\infty.$$

Although the optimization problem in (2.1) seems like a deterministic problem, the solution of the problem becomes stochastic due to the communication and computational noise when solving (2.1). In the application to wind farm control, the function f_i constitutes the cost of energy generation function for the ith wind turbine, the function F represents the cost of energy generation for the entire wind farm, and the computation noise v_{mi} is the stochastic wind.

2.3 LQR Problem Formulation

In the previous section, we formalized the optimization problem in the first level of our control scheme. In this section, we will formalize the optimization problem in the second level. At this level of control, we want to generate the controls for the individual wind turbines that are optimal for the operation of the entire wind farm to generate the maximum amount of energy with the least cost.

From Chapter 1, we see that the dynamics for the power control of the wind turbine are not linear. However, we see that often the dynamics can be linearized around the operating point of the wind turbine system through a small deviation argument [21,24]. Furthermore, different codes have been developed to model turbine dynamics using different methods. One approach is to use an assumed modes method to discretized the wind turbine structure. This method allows the modeling of the most important dynamics of the wind turbine with only a few degrees of freedom. One such dynamic code is the FAST dynamic code in [16]. This code has recently been modified to produce linear state-space models of turbine systems [36]. Therefore, motivated by both theory and simulation, we consider the dynamics of the wind turbine as a linear system.

Let us consider a linear time-invariant system that consists of a set $I = \{1,2,...,q\}$ of wind turbines or agents. We assume that these agents have ideal bidirectional communication with each other. For each $i \in I$, the state $x_i(k)$ belongs to $\mathbb{R}^n$ and the input $u_i(k)$ belongs to $\mathbb{R}^m$. The output y_i, which is the generated power, and the desired trajectory P, which captures the power demand, belongs to $\mathbb{R}$. We pose the cost of energy generation function for each wind turbine as a quadratic function of the state and the control effort. Since the cost function for the wind farm is the aggregate function of the wind turbine cost function, we wish to solve the optimization problem over a time horizon $\{1,2...,N\}$

$$\min_{u_1,\ldots,u_q} \sum_{i=1}^{q} \sum_{k=1}^{N-1} (x_i(k)'Q_i x_i(k) + u_i(k)'R_i u_i(k)) + x_i(N)'Q_i x_i(N) = J(x_1 \ldots x_n), \qquad (2.2)$$

where $Q_i \geq 0$, $R_i > 0$, $\forall i \in I$.

As mentioned in Chapter 1, a reason investigating a controller for the entire wind farm is the interaction of aerodynamic wakes between wind turbines. Taking aerodynamic wake interaction of the wind turbines into account, we consider a linear dynamically coupled model for the wind turbines in the wind farm by first defining the

following notations

$$x(k) = \begin{bmatrix} x_1(k) \\ \vdots \\ x_q(k) \end{bmatrix}, \quad u(k) = \begin{bmatrix} u_1(k) \\ \vdots \\ u_q(k) \end{bmatrix}, \quad C = \begin{bmatrix} C_1 \\ \vdots \\ C_q \end{bmatrix}$$
$$Q = \begin{bmatrix} Q_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & Q_q \end{bmatrix}, \quad R = \begin{bmatrix} R_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & R_q \end{bmatrix}. \tag{2.3}$$

Then, we can restate the coupled optimization problem and its constraints as

$$J(x_1 ... x_q) = \min_u \sum_{k=1}^{N-1} x(k)' Q x(k) + u(k)' R u(k)) + x(N)' Q x(N) \tag{2.4}$$
$$x(k+1) = Ax(k) + Bu(k), \quad A \in \mathbb{R}^{qn \times qn}, \quad B \in \mathbb{R}^{qn \times qm}, \tag{2.5}$$
$$Cx(k) = P(k), \quad C \in \mathbb{R}^{1 \times qn}. \tag{2.6}$$

The optimization problem (2.4) with the constraints (2.5) and (2.6) is called coupled Linear Quadratic Regulator (LQR) problem.

To tackle this coupled dynamics LQR problem, let us first consider the simpler decoupled dynamics LQR problem. We consider (2.2) with decoupled dynamics for each wind turbine. This is equivalent to ignoring all physical interactions between wind turbines, including aerodynamic wake interactions. Then, (2.2) is subject to the dynamics constraints

$$x_i(k+1) = A_i x_i(k) + B_i u_i(k), \quad A_i \in \mathbb{R}^{n \times n}, \quad B_i \in \mathbb{R}^{n \times m}, \tag{2.7}$$
$$y_i(k) = C_i x_i(k), \quad C_i \in \mathbb{R}^{1 \times n}, \quad \forall 1 \in I. \tag{2.8}$$

The optimization problem (2.2) together with the set of linear dynamics (2.7) and (2.8) forms the unconstrained LQR optimization problem. Finally, because the generated power must necessarily match the demand, we have the equality constraint

$$\sum_{i=1}^{q} y_i(k) = P(k). \tag{2.9}$$

Note that constraints (2.6) and (2.9) impose a coupling constraint between the outputs of the agents. This coupling constraint is one of the reasons for the need to find a different way to solve (4.1) than merely optimizing each individual cost function of each turbine.

In the next two chapters, we develop a stochastic control algorithm that solves the stochastic resource allocation formulation and a Tatonnement-based algorithm that solves the LQR formulation. Both of these algorithms are distributed algorithms intended for implementation at every wind turbine. Using these two algorithms, we have completely eliminated the centralized wind farm controller as seen in Figure 1.2. In-

stead, each wind turbine generates their own operating points and optimal controls as seen in Figure 2.1.

2.4 Figures

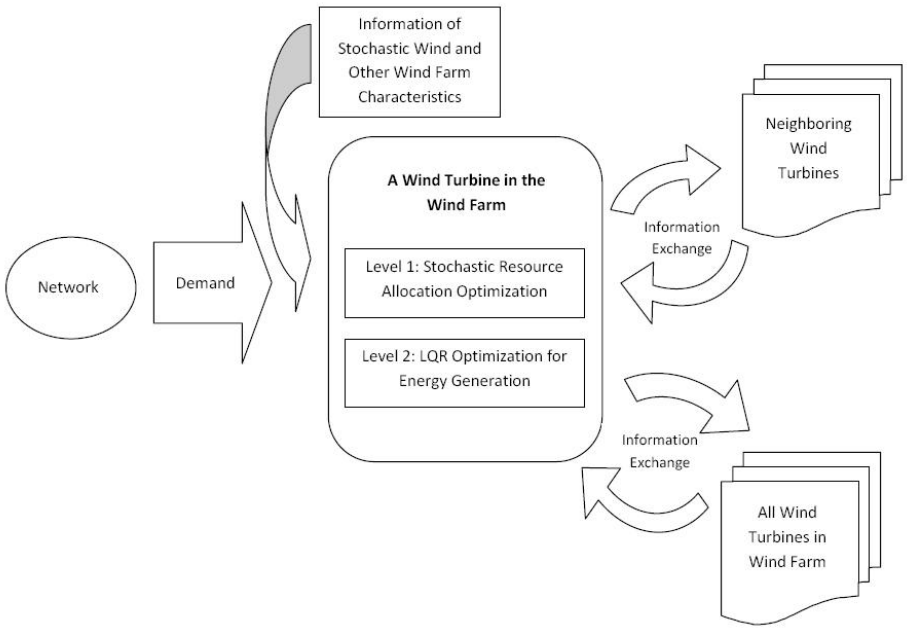

Figure 2.1: The proposed two level distributed control scheme for a wind turbine in the wind farm.

Chapter 3

Stochastic Control Algorithm

3.1 Introduction

In this chapter, we introduce a stochastic resource allocation algorithm, inspired by a deterministic resource allocation algorithm, that provides the solution to (2.1) almost surely. In doing so, it provides a method for each wind turbine to calculate its own operating points. The development of the algorithm is presented in the broader scope of network systems. Its application in the two level control scheme a wind farm is discussed in Chapter 2.

The algorithm developed in this section belongs to a more general class of stochastic gradient descent algorithms in [34]. Our algorithm is also related to algorithms in [27], [28], and [26]. In [27], the authors combine the incremental gradient algorithm [22] with Robbins Monro approximation method [30] to solve a stochastic optimization problem. This algorithm requires a ring structure in the network to pass information between nodes. In [28], each node randomly selects a neighbor and averages it's state with the selected neighbor. Then, each node adjusts the average along the negative direction of the gradient. The state is a scalar and is required to compute every agent's objective function. Finally, the algorithm in [27] updates the state with a sub-gradient evaluated at a weighted average of the states of its neighbors. This algorithm solves the problem where the entire state is needed to calculate each agent's objective function.

Our algorithm differs from these related algorithms in that we assume each agent can compute their objective function using only their local state. Also, instead of using projection to satisfy the constraint at each iteration, our algorithm uses a simpler mechanism through choosing an appropriate weighting matrix W that represents the communication structure of the network. Finally, our algorithm does not require the network to have a specific communication structure such as the ring structure in [27].

3.2 Stochastic Resource Allocation Algorithm

Many algorithms have been proposed to distributively solve convex optimization problems in a network environment (see, e.g., [13] for a survey). We focus on the problem of distributed convex optimization under an equality constraint, such as (3.1), which constitutes a standard model for resource allocation. In this context, it is often desir-

able for the equality constraint (which captures the fact that the amount of physically available resources is limited) to be satisfied at every iteration of the algorithm. This guarantees that the allocation obtained upon termination is always implementable, even if the nodes of the network are unable to run the algorithm to completion because of computational limitations or limited run-time requirements. An algorithm whose iterates satisfy the constraint at every time step is said to be "feasible", since it generates a sequence of feasible points for the underlying optimization problem.

Dual decomposition algorithms, which introduce the Lagrange multiplier associated with the quality constraint as a coordination variable among decoupled subproblems solved at each node, are *not* feasible since they only guarantee that the constraint is satisfied asymptotically [5, 9]. Primal decomposition algorithms are feasible, but require communication between all the nodes of the network if the equality constraint involves global coupling.

In contrast with these methods, Xiao and Boyd recently proposed a feasible distributed resource allocation algorithm for a noise-free network, which only requires communication between direct neighbors in the network [37]. This algorithm uses a very simple mechanism to guarantee feasibility at every iteration, and thus constitutes an attractive starting point for more complex wireless networks problems, in which computation and communication noises cannot be avoided [10, 18]. Since this algorithm does not converge in the presence of noise, we propose a slight modification, based on tools and ideas from the field of stochastic gradient approximation [3, 30], which results in a new algorithm, whose almost sure convergence to an optimal resource allocation can be established, in the presence of noise.

We start by reviewing some features of the Xiao-Boyd algorithm introduced in [37]. We then propose our extension to a stochastic algorithm . The analysis regarding effect of noise and convergence is analyzed. Finally, we illustrate the performance of the stochastic algorithm through a numerical example .

3.2.1 Algorithm Development

Recall the problem formulation in (2.1)

$$\min_x F(x) \quad s.t. \quad \hat{1}'x = c. \tag{3.1}$$

and Assumptions 1 and 2 in Chapter 2. We develop our stochastic algorithm base on the algorithm developed by [37]. We begin by reviewing the main properties of that algorithm. The algorithm in [37] is

$$x(k+1) = x(k) - W\nabla F(x(k)), \tag{3.2}$$

where $\nabla F(x)$ is defined as

$$\nabla F(x) = \begin{bmatrix} \frac{d}{dx_1} f_1(x_1) \\ \vdots \\ \frac{d}{dx_n} f_n(x_n) \end{bmatrix}. \tag{3.3}$$

The elements in the weighing matrix $W = [w_{ij}]$ have the properties

$$\begin{aligned} w_{ij} &= 0 \quad \forall j \notin N_i \\ w_{ij} &\neq 0 \quad \forall j \in N_i \\ w_{ii} &\neq 0 \quad \forall i \in \{1, \cdots, n\}. \end{aligned}$$

In addition, W is assumed to satisfy

$$\hat{1}'W = 0 \tag{3.4}$$

$$W\hat{1} = 0. \tag{3.5}$$

Property (3.4) is used to ensure that algorithm (3.2) satisfies the feasibility condition

$$\hat{1}'x(k+1) = \hat{1}'x(k) = c \quad \forall k \tag{3.6}$$

since

$$\hat{1}'x(k+1) = \hat{1}'x(k) - \hat{1}'W\nabla F(x(k)) = \hat{1}'x(k), \forall k. \tag{3.7}$$

On the other hand, property (3.5) guarantees that the optimal solution of (3.1), x^*, is a stationary point of algorithm (3.2) since

$$x^* = x^* - W(\lambda^*\hat{1}) = x^*. \tag{3.8}$$

The fact that $\nabla F(x^*) = \lambda^*\hat{1}$ follows from the KKT conditions [5]

$$\nabla F(x^*) = \lambda^*\hat{1}, \text{ for a unique } \lambda^* \in \mathbb{R}, \tag{3.9}$$

$$\hat{1}'x^* = c. \tag{3.10}$$

Under these assumptions on W and an additional assumption on the eigenvalues of W, Xiao and Boyd proved in [37] that algorithm (3.2) converges to the optimal solution of (3.1) in a directed network with no noise.

We cannot directly apply algorithm (3.2) to problem (3.1) in our network because the implicit assumption of perfect gradient information in algorithm (3.2) is not valid in a network with computation and communication noise. Recall that the computation

noise, v_{m_i}, and the communication noise, $v_{c_{j\to i}}$, is defined as

$$\begin{aligned} v_{m_i} &: \Omega \times \mathbb{N} \to \mathbb{R} \quad \forall i \in \{1, \cdots, n\} \\ v_{c_{j\to i}} &: \Omega \times \mathbb{N} \to \mathbb{R} \quad \forall i, j \in \{1, \cdots, n\}, \quad i \neq j. \end{aligned}$$

and the noise processes are defined respectively as

$$\begin{aligned} [v_{c_{j\to i}}(k)]_{(\omega)} &= v_{c_{j\to i}}(\omega, k). \quad \forall \omega \in \Omega \\ [v_{m_i}(k)_{(\omega)}] &= v_{m_i}(\omega, k) \quad \forall \omega \in \Omega \end{aligned}$$

for all k. In the application of wind farm control, v_{m_i} represents the stochastic wind.

Taking inspiration from algorithm (3.2), we propose the following extension for each agent i in our network

$$x_i(k+1) = x_i(k) - \gamma(k) w_{ii} \left(\frac{d}{dx_i} f_i(x_i) + v_{m_i}(k) \right) \tag{3.11}$$

$$- \gamma(k) \sum_{\substack{i \in N_j \\ j \neq i}} w_{ij} \left(\frac{d}{dx_j} f_j(x_j) + v_{m_j}(k) + v_{c_{j\to i}}(k) \right). \tag{3.12}$$

The sequence $\{\gamma(k)\}_{k\in\mathbb{N}}$ is monotonically decreasing with the properties

$$\gamma(k) > 0 \quad \forall k \tag{3.13}$$

$$\sum_{k=1}^{\infty} \gamma(k) = \infty \tag{3.14}$$

$$\sum_{k=1}^{\infty} \gamma(k)^2 < \infty. \tag{3.15}$$

Such a step is standard in Robbins-Monro type stochastic gradient algorithms [30] and is a typical way to ensure convergence. To simplify notation, we rewrite algorithm (3.12) as

$$x(k+1) = x(k) - \gamma(k) \left(W r(k) + v_c(k) \right), \tag{3.16}$$

where

$$v_m(k) = \begin{bmatrix} v_{m_1}(k) \\ \vdots \\ v_{m_n}(k) \end{bmatrix}, \quad v_c(k) = \begin{bmatrix} \sum_{1\in N_j} w_{1j} v_{c_{j\to 1}}(k) \\ \vdots \\ \sum_{n\in N_j} w_{nj} v_{c_{j\to n}}(k) \end{bmatrix}, \quad r(k) = \nabla F(x(k)) + v_m(k).$$

Note that if $i \neq j$, the random variables $r_i(k)$ and $r_j(k)$ can be correlated if the network graph contains a node l with the property that there exist directed paths of length less than k between l and i, and l and j.

Let us define

$$\Gamma(k) = [x(1)\cdots x(k), v_m(1)\cdots v_m(k-1)]'$$

and denote the expectation conditioned on $\Gamma(k)$ as $\hat{\mathbb{E}}_k\{.\} = \mathbb{E}\{.|\Gamma(k)\}$. We note that

$$\hat{\mathbb{E}}_k\{\nabla F(x(k)) + v_m(k)\} = \hat{\mathbb{E}}_k\{\nabla F(x(k))\} + \hat{\mathbb{E}}_k\{v_m(k)\} = \nabla F(x(k)), \tag{3.17}$$

where the last equality comes from the zero mean assumption in Assumption 1 and the definition of $\Gamma(k)$. In the next section, we analyze the effects of noise and show that under certain conditions, algorithm (3.16) converges almost surely to the optimal value of problem (3.1).

3.3 Algorithm Properties

3.3.1 Noise Effects

We analyze the effects of computation and communication noise on the properties of algorithm (3.16). In the process, we justify retaining the property

$$\hat{1}'W = 0 \tag{3.18}$$

when choosing weight matrix W in algorithm (3.16).

Ideally, we would like algorithm (3.16) to be feasible in the sense that constraint (3.6) should hold. While this is not possible because of the noise, assuming that W satisfies property (3.18) allows us to prove that algorithm (3.16) retain a property analogous to equation (3.7), namely:

$$\mathbb{E}\{\hat{1}'x(k+1)\} = \mathbb{E}\{\hat{1}'x(k)\} + \gamma(k)\left(\mathbb{E}\{\hat{1}'v_c(k)\} + \hat{1}'W\mathbb{E}\{\nabla F(x(k))\}\right) = \mathbb{E}\{\hat{1}'x(k)\}. \tag{3.19}$$

Furthermore, provided that one can establish the convergence of sequence $\{x(k)\}$ to $\hat{x}$ (a point we will tackle shortly), it can also be shown that the variance of $\hat{1}'\hat{x}$ is bounded if there exists $\beta \in \mathbb{R}^+$ such that $Var(v_{c_i}(k)) \leq \beta$ for all i. From Assumption 1, we have

$$\begin{aligned} Var(\hat{1}'x(k+1)) &= Var(\hat{1}'x(k) + \gamma(k)\hat{1}'v_c(k)) = Var(\hat{1}'x(k)) + \gamma(k)^2 Var(\hat{1}'v_c(k)) \\ &\leq n\beta \sum_{q=1}^{k} \gamma(q)^2, \end{aligned} \tag{3.20}$$

where n is the number of agents in the network. Using the fact that $\{\gamma(k)\}$ is square-summable, as assumed in (3.15), yields

$$\lim_{k\to\infty} Var(\hat{1}'x(k)) = Var(\hat{1}'\hat{x}) \leq n\beta \sum_{q=1}^{\infty} \gamma(q)^2 < \infty. \tag{3.21}$$

From the upper bound on the variance of the equality constraint (3.20), we see that the variance of the equality constraint at each iteration is a function of the *communication noise* only. It follows that algorithm (3.16) is feasible when computation noise is the only one present, and W satisfies (3.18).

3.3.2 Convergence Analysis

We have shown the effects of the computation and communication noise on properties of algorithm (3.16). Next, we prove that if sequence $\{x(k)\}$ is generated according to (3.16), $\{F(x(k)\}$ converges almost surely to a random variable F_∞. In the absence of communication noise, we show in addition that F_∞ is the optimal value of problem (3.1), and that the sequence $\{x(k)\}$ converges almost surely to x^*, its unique optimal point. Our proof relies on a classical tool for the analysis of stochastic gradient algorithms called the Quasi-Martingale Convergence Theorem [3, 9]. We start by recalling the Quasi-Martingale Convergence Theorem, as presented in [20], in Lemma 1.

Lemma 1. *Let $(\hat{\Omega}, \hat{\mathscr{F}}, \hat{P})$ be a probability space. Let $(\hat{\mathscr{F}}(k))_{k\in\mathbb{N}}$ be an increasing family of sub-σ-algebras of $\hat{\mathscr{F}}$ such that $\cup_{k\in\mathbb{N}}\hat{\mathscr{F}}(k) = \hat{\mathscr{F}}$. Let $(Y(k))_{k\in\mathbb{N}}$ be a real valued stochastic process such that $Y(k)$ is $\hat{\mathscr{F}}(k)$ measurable for every k. For every k, let the event $G(k)$ be defined as*

$$G(k) := \{\mathbb{E}\{Y(k+1) - Y(k) | \hat{\mathscr{F}}(k)\} > 0\}$$

and define the indicator function $\bar{1}_{G(k)}$ as

$$\bar{1}_{G(k)} = \begin{cases} 1 & \text{if } G(k) \text{ is true} \\ 0 & \text{otherwise} \end{cases} \tag{3.22}$$

Finally, let us assume that

$$\sum_{k=1}^{\infty} \mathbb{E}\{\bar{1}_{G(k)} \cdot (Y(k+1) - Y(k))\} < \infty \tag{3.23}$$

$$\liminf_k \mathbb{E}\{Y(k)\} > -\infty \tag{3.24}$$

$$\sup_k \mathbb{E}\{Y(k)^-\} < \infty, \tag{3.25}$$

where $Y(k)^- = -\min\{0, Y(k)\}$. Then the sequence $(Y(k))_{k\in\mathbb{N}}$ converges to a integrable random variable Y_∞ almost surely.

Proof. See Theorem 9.4 and Proposition 9.5 in page 49-51 of [20]. □

Theorem 1. *Let Assumption 1 and Assumption 2 hold, and let W be a square matrix with the following properties:*

$$1'W = 0, \tag{3.26}$$

$$\lambda_{min} > 0, \text{ and } W' = W, \tag{3.27}$$

where λ_{min} is defined as the second smallest *eigenvalue of W. Assume that there exist $\alpha_1, \alpha_2, \alpha_3, \beta \in \mathbb{R}^+$ such that*

$$Var(r_i(k)) \leq \alpha_1, \quad \mathbb{E}\{r_i(k)\}^2 \leq \alpha_2^2, \quad \mathbb{E}\{r_i(k)^2\} \leq \alpha_3 \quad \forall i,k, \tag{3.28}$$

where $r_i(k)$ designates the i-th element of vector $r(k)$, and

$$Var(v_{c_i}(k)) < \beta \quad \forall i,k. \tag{3.29}$$

Then $F(x(k))$ converges almost surely to some random variable F_∞. If there is no communication noise, then $F_\infty = F^*$, the optimal value of problem (3.1)*. In addition, $x(k)$ converges to x^* almost surely, with x^* being unique optimal point of problem* (3.1)*.*

The central idea for the proof of Theorem 4 is to show that the process $F(x(k))$ satisfies all of the assumptions in Quasi-Martingale Convergence Theorem, thereby proving that $F(x(k))$ is a quasi-martingale process and converges almost surely. Furthermore, in the absence of communication noise, we show that $\nabla F(x(k))$ converges almost surely to $\lambda^* \hat{1}$ for some $\lambda^* \in \mathbb{R}$, which means that the KKT conditions are satisfied almost surely, and proves that $\hat{x} = x^*$ is the optimal value of problem (3.1).

Proof. From Taylor's Theorem of Remainders, we know that, for every i and k, there exists $z_i(k) \in (x_i(k), x_i(k+1))$ such that

$$\begin{aligned} f_i(x_i(k+1)) = f_i(x_i(k)) &+ \frac{d}{dx_i} f_i(x_i(k))(x_i(k+1) - x_i(k)) \\ &+ \frac{1}{2}\frac{d^2}{dx_i^2} f_i(z_i(k))(x_i(k+1) - x_i(k))^2. \end{aligned}$$

Let $\Delta x(k) = x(k+1) - x(k)$. Summing over i, we have

$$\begin{aligned} F(x(k+1)) &= F(x(k)) + \sum_{i=1}^{n} \frac{d}{dx_i} f_i(x_i(k))(x_i(k+1) - x_i(k)) \\ &\quad + \frac{1}{2}\sum_{i=1}^{n} \frac{d^2}{dx_i^2} f_i(z_i(k))(x_i(k+1) - x_i(k))^2 \\ &= F(x(k)) + \nabla F(x(k))' \Delta x(k) + \frac{1}{2}\Delta x(k)' \nabla^2 F(z(k)) \Delta x(k), \end{aligned}$$

where ∇F is defined in (3.3), $\nabla^2 F$ is the (diagonal) Hessian matrix of F, and $z(k)$ is defined as

$$z(k) = \begin{bmatrix} z_1(k) \\ \vdots \\ z_n(k) \end{bmatrix}.$$

From the upper bound assumption on f_i's second derivative detailed in Assumption

2, there exists a diagonal matrix U such that $U \geq \nabla^2 F(x), \forall x$. Consequently, since

$$\Delta x(k) = -\gamma(k)\left(Wr(k) + v_c(k)\right),$$

we have the inequality

$$\begin{aligned} F(x(k+1)) \leq F(x(k)) &- \gamma(k)\nabla F(x(k))'Wr(k) - \gamma(k)\nabla F(x(k))'v_c(k) \\ &+ \frac{1}{2}\gamma(k)^2 r(k)W'UWr(k) + \gamma(k)^2 r(k)'W'Uv_c(k) \\ &+ \frac{1}{2}\gamma(k)^2 v_c(k)'Uv_c(k). \end{aligned} \tag{3.30}$$

Taking the expectation of (3.30), we obtain

$$\begin{aligned} \mathbb{E}\{F(x(k+1))\} \leq \mathbb{E}\{F(x(k))\} &- \gamma(k)\mathbb{E}\{\nabla F(x(k))'Wr(k)\} - \gamma(k)\mathbb{E}\{\nabla F(x(k))'v_c(k)\} \\ &+ \frac{1}{2}\gamma(k)^2\mathbb{E}\{r(k)'W'UWr(k)\} + \gamma(k)^2\mathbb{E}\{r(k)'W'Uv_c(k)\} \\ &+ \frac{1}{2}\gamma(k)^2\mathbb{E}\{v_c(k)'Uv_c(k)\}. \end{aligned} \tag{3.31}$$

Let us analyze each term of the expectation inequality (3.31). For the second term of (3.31), from the property that the expectation of $r(k)$ equals $\nabla F(x(k))$ in (3.17), we have

$$\begin{aligned} \mathbb{E}\{\nabla F(x(k))'Wr(k)\} &= \mathbb{E}\{\hat{\mathbb{E}}_k\{\nabla F(x(k))'Wr(k)\}\} = \mathbb{E}\{\nabla F(x(k))'W\hat{\mathbb{E}}_k\{r(k)\}\} \\ &= \mathbb{E}\{\nabla F(x(k))'W\nabla F(x(k))\}. \end{aligned} \tag{3.32}$$

Let us orthogonally decompose $\nabla F(x(k))$ as

$$\nabla F(x(k)) = p(k) + e(k),$$

where $p(k)$ denotes the component of $\nabla F(x(k))$ that is parallel to the vector $\hat{1}$, and $e(k)$ denotes the component lying in the hyperplane perpendicular to $\hat{1}$. More specifically, we define $e(k)$ as

$$e(k) = \nabla F(x(k)) - \frac{\hat{1}\nabla F(x(k))}{\hat{1}'\hat{1}}\hat{1}. \tag{3.33}$$

Using the definition of $e(k)$ in (3.33) with assumptions (3.26) and (3.27) on matrix W, we can rewrite equality (3.32) as

$$\mathbb{E}\{\nabla F(x(k))'Wr(k)\} = \mathbb{E}\{e(k)'W'e(k)\} \geq \lambda_{min}\mathbb{E}\{e(k)'e(k)\}.$$

Let us define $||.||$ as the Euclidean norm of a vector. Since $\mathbb{E}\{e(k)'e(k)\} \geq ||\mathbb{E}\{e(k)\}||^2$, we have the inequality

$$\mathbb{E}\{\nabla F(x(k))'Wr(k)\} \geq \lambda_{min}||\mathbb{E}\{e(k)\}||^2. \tag{3.34}$$

For the third and fifth terms of expectation inequality (3.31), we see from algorithm (3.16) and the independence assumptions in Assumption 1 that $v_{c_j}(k)$ is independent of $x_i(k)$for any $i, j \in \{1, \cdots, n\}$. Since $v_m(k)$ is also independent of $v_c(k)$, we have that $r(k)$ is independent of $v_c(k)$ as well. Therefore, from the zero mean assumption in Assumption 1, we have

$$\mathbb{E}\{\nabla F(x(k))' v_c(k)\} = \mathbb{E}\{\nabla F(x(k))\}' \mathbb{E}\{v_c(k)\} = 0 \tag{3.35}$$

and

$$\mathbb{E}\{r(k)' W U v_c(k)\} = tr\left(WU\mathbb{E}\{v_c(k) r(k)'\}\right) = tr\left(WU\mathbb{E}\{v_c(k)\}\mathbb{E}\{r(k)\}'\right) = 0, \tag{3.36}$$

where $tr(.)$ denotes the trace of a matrix.

To analyze the fourth term in (3.31), let us define $Cov(.)$ as the covariance function. From assumption (3.28) on the variance of $r(k)$, we have

$$|Cov(r_i(k), r_j(k))| \leq \sqrt{Var(r_i(k)) Var(r_j(k))} \leq \alpha_1, \quad \forall i, j, i \neq j.$$

Hence

$$|\mathbb{E}\{r_i(k) r_j(k)\}| \leq |Cov(r_i(k), r_j(k))| + |\mathbb{E}\{r_i(k)\}\mathbb{E}\{r_j(k)\}| \leq \alpha_1 + \alpha_2^2 \quad \forall i, j, i \neq j. \tag{3.37}$$

Let $\hat{\alpha}_1 \in \mathbb{R}^+$ be such that

$$tr((W'UW)^2) < \hat{\alpha}_1. \tag{3.38}$$

By assumption (3.28) on the variance on $r(k)$ and the upper bound on $|\mathbb{E}\{r_i(k) r_j(k)\}|$ in inequality (3.37), there exists $\hat{\alpha}_2 \in \mathbb{R}^+$ such that

$$tr((\mathbb{E}\{r(k) r(k)'\})^2) < n\alpha_3^2 + n(n-1)(\alpha_1 + \alpha_2^2)^2 < \hat{\alpha}_2. \tag{3.39}$$

From the fact that $W'UW$ is a semi-positive definite matrix, we see that $\mathbb{E}\{r(k) W'U'W r(k)\} \geq 0$. If we define $\hat{\alpha} = \max(\hat{\alpha}_1, \hat{\alpha}_2)$, we can use the inequalities (3.38) and (3.39), the definition of $\hat{\alpha}$, and Cauchy-Schwarz inequality to upper bound the fourth term as

$$\begin{aligned} \mathbb{E}\{r(k) W'U'W r(k)\} &= |tr(W'UW\mathbb{E}\{r(k) r(k)'\})| \\ &\leq \sqrt{tr(WU'W^2) tr(\mathbb{E}\{r(k) r(k)'\}^2)} < \hat{\alpha} \end{aligned} \tag{3.40}$$

Finally, let σ be the largest element in U. Then, using assumption (3.29) on the variance of $v_{c_i}(k)$, the zero mean assumption of $v_{c_i}(k)$ in Assumption 1, and the fact that U is a diagonal matrix, we upper bound the last term of expectation inequality

(3.31) as

$$\mathbb{E}\{v_c(k)'Uv_c(k)\} \leq \sigma \sum_{i=1}^{n} Var(v_{c_i}(k)) \leq n\sigma\beta. \tag{3.41}$$

Let us define

$$\Delta F(x(k)) = F(x(k+1)) - F(x(k)) \tag{3.42}$$

$$\kappa = \frac{1}{2}\hat{\alpha} + \frac{1}{2}n\sigma\beta. \tag{3.43}$$

Combining the equalities (3.35) and (3.36), which describes the third and fifth terms of the expectation inequality (3.31), with inequalities (3.40) and (3.41), which describes the second and fourth terms of the expectation inequality (3.31), and using definitions (3.42) for $\Delta F(x(k))$ and (3.43) for κ, we conclude that

$$\mathbb{E}\{\Delta F(x(k))\} \leq -\gamma(k)\lambda_{min}||\mathbb{E}\{e(k)\}||^2 + \kappa\gamma(k)^2 \leq \kappa\gamma(k)^2. \tag{3.44}$$

Now, we introduce the event $H(k)$ as

$$H(k) := \{\mathbb{E}\{(F(x(k+1)) - F(x(k))|\Gamma(k)\} > 0\}.$$

and it's indicator function as

$$1_{H(k)} = \begin{cases} 1 & \text{if } H(k) \text{ is true} \\ 0 & \text{otherwise.} \end{cases}$$

Then, the upper bound on $\mathbb{E}\{\Delta F(x(k))\}$ in (3.44) yields

$$\mathbb{E}\{1_{H(k)} \cdot \Delta F(x(k))\} \leq \kappa\gamma(k)^2. \tag{3.45}$$

Taking the infinite sum of the upper bound in (3.45) on both sides yields

$$\sum_{k=1}^{\infty} \mathbb{E}\{1_{H(k)} \cdot \Delta F(x(k))\} \leq \kappa \sum_{k=1}^{\infty} \gamma(k)^2 < \infty, \tag{3.46}$$

where the last inequality follows from the square summable property of $\gamma(k)$ in (3.15). We see from the upper bound of the infinite summation in (3.46) that (3.23) in Lemma 3 is satisfied. Also, since F is a non-negative function by Assumption 2, conditions (3.24) and (3.25) of Lemma 3 are satisfied. Consequently, we conclude by Lemma 1 that the process $F(x(k))$ converges almost surely.

Now, let us consider the case with no communication noise. Then, inequality (3.30) becomes

$$F(x(k+1)) \leq F(x(k)) - \gamma(k)\nabla F(x(k))'Wr(k) + \frac{1}{2}\gamma(k)^2 r(k)W'UWr(k). \tag{3.47}$$

We take the expectation of inequality (3.47) conditioned on $\Gamma(k)$ and recall the defini-

tion of $\hat{\mathbb{E}}_k\{.\}$ as used in equation (3.17). We then obtain

$$\hat{\mathbb{E}}_k\{\Delta F(x(k))\} \leq -\gamma(k)\nabla F(x(k))'W\hat{\mathbb{E}}_k\{r(k)\} + \frac{1}{2}\gamma(k)^2\hat{\mathbb{E}}_k\{r(k)'W'UWr(k)\} \quad (3.48)$$

From the fact that the conditional expectation of $r(k)$ equals $\nabla F(x(k))$ in (3.17), we have

$$\nabla F(x(k))'W\hat{\mathbb{E}}_k\{r(k)\} = \nabla F(x(k))'W\nabla F(x(k)). \quad (3.49)$$

Using definition (3.33) of $e(k)$ and assumptions (3.26) and (3.27) on matrix W, we can rewrite equality (3.49) as

$$\nabla F(x(k))'W\nabla F(x(k)) = e(k)'W'e(k) \geq \lambda_{min}e(k)'e(k). \quad (3.50)$$

From the upper bound on $\mathbb{E}\{r(k)W'UWr(k)\}$ in (3.40), we concluded that there exists $\bar{\alpha}$ such that

$$\hat{\mathbb{E}}_k\{r(k)WUWr(k)\} \leq \bar{\alpha} < \infty. \quad a.s. \quad (3.51)$$

Using the lower and upper bounds in (3.50) and (3.51), we upper bound the inequality (3.48) almost surely as

$$\hat{\mathbb{E}}_k\{\Delta F(x(k))\} \leq -\lambda_{min}e(k)'e(k) + \gamma(k)^2\hat{\alpha} \quad a.s. \quad (3.52)$$

Since $F(x(k))$ converges almost surely, we conclude that

$$\lambda_m\sum_{k=1}^{\infty}\gamma(k)\|e(k)\|^2 < \infty \quad a.s.$$

because $\{\gamma(k)\}$ is square-summable. This implies that there exists a set of cluster points for the series $e(k)$ and a corresponding set of cluster points for $x(k)$. Furthermore, since $\gamma(k)\|e(k)\|^2 \geq 0$, it follows that $\liminf_{k\to\infty}\|e(k)\|^2 = 0$ almost surely.

Let $\{e(\phi(k))\}$ be a subsequence of $\{e(k)\}$ that converges to zero almost surely. Let $\{x(\phi(k))\}$ be the corresponding subsequence extracted from $\{x(k)\}$. Since we know that $F(x(k))$ converges almost surely to F_∞, $\lim_{k\to+\infty}F(x(\phi(k))) = F_\infty$ as well, which implies that the sequence $\{F(x(\phi(k)))\}$ is bounded. By the positivity of each function f_i, this implies that $f_i(x_i(\phi(k)))$ is bounded for all k and i. Invoking the coercivity of each f_i, we deduce that each sequence $\{x_i(\phi(k))\}$ $(i = 1,...,N)$ is bounded for all k and, hence, admits a convergent subsequence. Then, using a diagonal argument, we can find an increasing map $\psi : \mathbb{N} \to \mathbb{N}$ such that $\{x_i(\psi(\phi(k)))\}$ converges for all i, i.e., such that the vector-valued sequence $\{x(\psi(\phi(k)))\}$ converges to some $\hat{x}$. By the convergence of $\{e(\phi(k))\}$ there exists $\lambda^* \in \mathbb{R}^+$ such that $\lambda^*\hat{1} = \lim_{k\to\infty}\nabla F(x(\psi(\phi(k))))$ almost surely. Then, by continuity of ∇F from Assumption 2, we have

$$\lambda^*\hat{1} = \nabla F(\hat{x}) \text{ almost surely.} \quad (3.53)$$

In other words, $\hat{x}$ satisfies the KKT conditions. But, since the domain of F is open and we only have a linear equality constraint, (3.1) satisfies the Slater's Condition, i.e., the KKT conditions are necessary and sufficient. As a result, (3.53) implies that $\hat{x}$ is the unique minimum of F and, thus, $F(\hat{x}) = F^*$.

Because we have already proved that $F(x(k))$ converges to F_∞ almost surely, we know that every subsequence of $F(x(k))$ must also converge to the same F_∞ almost surely. Since we have just showed that one of the subsequences converges to F^*, we conclude that $F_\infty = F^*$. From strict convexity of F, the x^* associated with F^* is also the unique global minimum. Therefore, every subsequence of $x(k)$ converges to the same point x^* almost surely, and it follows that $x(k)$ converges to x^* almost surely. □

We end by noting that the results of Theorem 4 are still valid if we consider $r(k)$ as a more general function of the form

$$r(k) = \begin{bmatrix} r_1(x_1(k), v_{m_1}(k)) \\ \vdots \\ r_n(x_n(k), v_{m_n}(k)) \end{bmatrix}.$$

with the property

$$\hat{\mathbb{E}}_k\{r(k)\} = \nabla F(x(k)) \quad \forall k.$$

3.3.3 Numerical Simulation

We compare the performance of algorithm (3.16) with algorithm (3.2) for agents with the same objective functions as those in the numerical example of [37]. We consider a 3-regular strongly connected directed graph with $n = 6$ agents (instead of $n = 20$ as in [37] for the sake of simulation run time). The topology of the network can be inferred from the structure of the symmetric weight matrix

$$W = \frac{1}{25}\begin{bmatrix} 3 & -1 & 0 & -1 & 0 & -1 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 3 & -1 & 0 & -1 \\ -1 & 0 & -1 & 3 & -1 & 0 \\ 0 & -1 & 0 & -1 & 3 & -1 \\ -1 & 0 & -1 & 0 & -1 & 3 \end{bmatrix}.$$

For the simulation, the following family of non-negative functions is used:

$$f_i(x_i) = \frac{1}{2}a_i(x_i - c_i)^2 + \log[1 + e^{b_i(x_i - d_i)}], \tag{3.54}$$

for all i, with coefficients a_i, b_i, c_i, d_i generated randomly according to the uniform distribution on intervals $[0,2]$, $[-2,2]$, $[-10,10]$, $[-10,10]$ respectively. This family of

functions has the second derivative

$$\frac{d^2}{dx_i^2} f_i(x_i) = a_i + \frac{b_i^2 e^{b_i(x_i - d_i)}}{(1 + e^{b_i(x_i - d_i)})^2} > 0 \quad \forall x_i$$

with the upper bounds $a_i + \frac{1}{4} b_i^2$.

Let us define each component of vector $r(k)$ as

$$r_i(k) = \frac{d}{dx_i} f_i(x_i) + v_{m_i}(k).$$

The values for $v_{m_i}(k)$ and $v_{c_i}(k)$ are generated independently by a Gaussian pseudo-random number generator with a mean of 0 and a variance of 3. With no loss of generality, we assume that the resource constraint takes the form

$$\sum_{i=1}^{n} x_i = 0. \tag{3.55}$$

and choose the initial conditions

$$x(0) = \begin{bmatrix} 0.1 & -0.1 & 0.1 & -0.1 & 0.1 & -0.1 \end{bmatrix}.$$

In the remainder of this section, we will say that "$F(x(k))$ converges to a value with ε" if $F(x(k))$ satisfies

$$|f_i(x_i(k+1)) - f_i(x_i(k))| < \varepsilon, \text{ for all } i. \tag{3.56}$$

Note that this does not necessarily mean that $x(k)$ itself converges.

We ran multiple trials to observe the performance of algorithm (3.16) over a sufficiently large sample. Using a pseudo-random number generator in MATLAB, we generated the following vectors of coefficients

$$a = \begin{bmatrix} 0.4518 \\ 0.6222 \\ 1.8098 \\ 0.5161 \\ 1.2057 \\ 0.5934 \end{bmatrix}, \quad b = \begin{bmatrix} -1.3172 \\ 1.6935 \\ 1.9190 \\ -0.3651 \\ 0.8449 \\ -0.7249 \end{bmatrix}, \quad c = \begin{bmatrix} -5.4467 \\ -1.3959 \\ -1.2226 \\ 1.8979 \\ -5.5651 \\ -1.5167 \end{bmatrix}, \quad d = \begin{bmatrix} -1.2860 \\ -6.3037 \\ -7.7776 \\ -4.7558 \\ -7.6516 \\ 0.1572 \end{bmatrix}.$$

For each trial, we first applied algorithm (3.2) to the objective functions in (3.54) until algorithm (3.2) converged to $\hat{x}^*$ with $\varepsilon = 0.001$. This convergence is guaranteed in [37]. We set $F(\hat{x}^*) = F^*$ as the benchmark for the performance of algorithm (3.16). Then, we applied algorithm (3.16) to the objective functions in (3.54) until $F(x(k))$ converged to a value F_∞ with $\varepsilon = 0.001$. We then computed the error $F_\infty - F^*$.

We first simulated the case with no communication error. Since every function f_i in (3.54) is strictly convex, we expect algorithm (3.16) to converge to x^* almost surely by Theorem 4. In this simulation, we used the step $\gamma(k) = \frac{15}{k}$. The result is plotted

in Figure 3.1. The empiricial mean of the error of algorithm (3.16) for 1000 trials is 1.03×10^{-2}, while its empirical variance is 5.2629×10^{-5}. Note that since $x(k)$ is feasible for all k because of the absence of communication noise, $F(x(k)) \geq F^*$ for all k. In particular, $F_\infty \geq F^*$. This is in accordance with Figure 3.1.

In the presence of communication noise, we ran 10000 trials with the step $\gamma(k) = \frac{7}{10k}$. According to Theorem 4, we expect convergence of $F(x(k))$ to some random variable F_∞ in this case, but do *not* expect that F_∞ equals F^*. We also expect the empirical mean of $\hat{1}'\hat{x}$ to be small. Results are plotted in Figure 3.2. The empirical mean of the error of algorithm (3.16) for this simulation is 13.0261, while the variance is 205.5396. We also see that the mean of $\hat{1}'\hat{x}$ is 0.0045277 with a variance of 42.9357.

In contrast with our observations on Figure 3.1, note that when communication noise is active, the feasibility condition only holds in expectation at every k. Hence on a particular realization, $x(k)$ may not be feasible for problem (3.1), and it is thus possible that $F_\infty < F^*$. This can be seen to occur on Figure 3.2. From the simulation, the performance of algorithm (3.16) coincides with theory. Algorithm (3.16) performs well without communication noise when compared to algorithm (3.2). However, with communication error, we cannot guarantee that the algorithm reaches the optimum.

The error from optimum in the case of no communication errors can be attributed to a couple of factors. If the magnitude of the ε in definition of (3.56) is smaller, then the error of (3.16) decreases but the runtime of the simulation increases. Also, the fact that the noise is only pseudo-random may play a role in the deviation of the performance of algorithm (3.16) from algorithm (3.2).

3.4 Figures

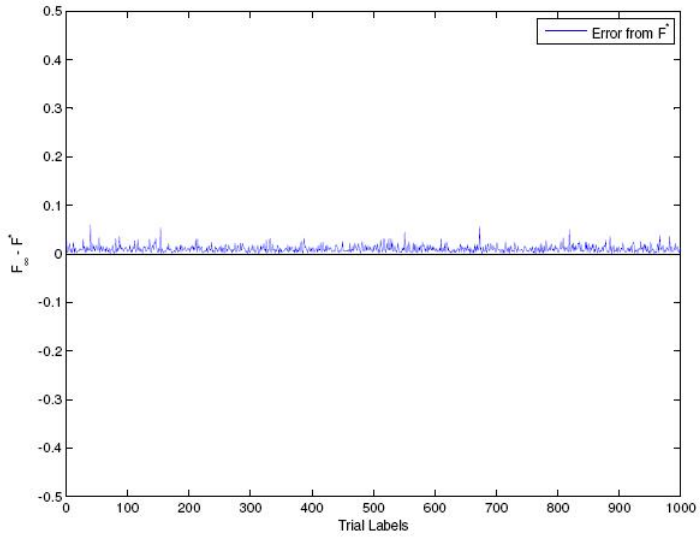

Figure 3.1: The error defined by the difference of F_∞ generated by algorithm (3.16) and F^* with $\gamma(k) = \frac{15}{k}$ and no communication noise.

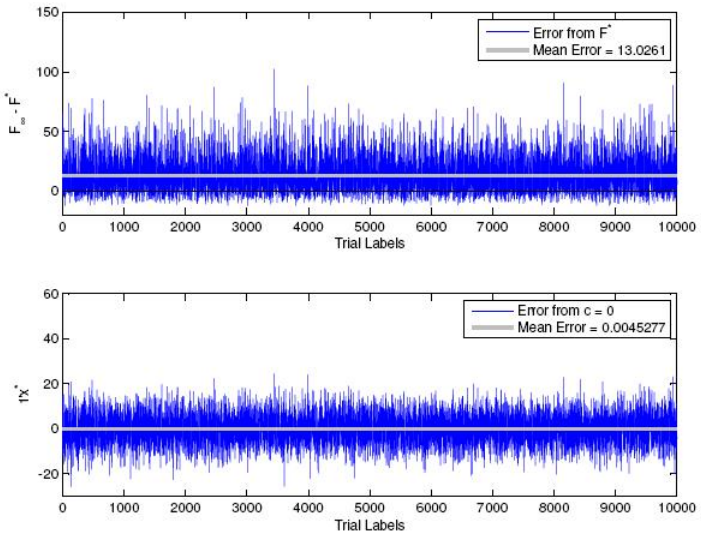

Figure 3.2: Top: The error defined by the difference of F_∞ generated by algorithm (3.16) and F^* with $\gamma(k) = \frac{7}{10k}$ and communication noise. Bottom: The error defined by the difference between $\hat{1}'x^*$ and the equality constraint $c = 0$.

Chapter 4

Tatonnement-based Control Algorithm

4.1 Introduction

In this chapter, we introduce a decentralized algorithm to solve the LQR optimization problem posed in Chapter 2 for both a dynamically decoupled and coupled system. This algorithm is used in the second level of our control scheme to generate the optimal controls for each wind turbine's energy generation. We call our algorithm "Tatonnement-Based" because it is inspired by the Tatonnement process in Economics. It is also similar to the so-called dual/price decomposition approach, such as [29]. The Tatonnement process was first introduced by Walras in 1874. This process was intended to model the effects of the free market trade. The process of Tatonnement begins with an auctioneer announcing a price for some goods to a group of auctionees. The auctionees respond with their demands at that price. The auctioneer either lower the price if there is too little demand or raises it if there is too much. Then, he announces the new price to the auctionees, and the process repeats until an equilibrium price is reached such that all the demands of the auctionees are met. Trading between auctioneer and the auctionees occurs only after this equilibrium is achieved [6, 35]. As a preview, in the perspective of our algorithm, this price is the Lagrange multiplier associated with the equality constraint of the problem, and the demand is the state or the control effort of each agent as a function of this Lagrange multiplier.

Our Tatonnement-based algorithm uses ideas from decentralized LQR control and price control. The idea of decentralized LQR controller for decoupled dynamic systems is investigated by [2]. The authors in [2] propose a control design method that exploits the decoupling nature of the feedback matrix K for LQR problems in the environment of decoupled dynamic systems. However, an important difference between the problem formulated by [2] and our problem is that [2] does not have any constraints on the states other than the dynamic constraints, while our problem has an equality constraint.

The idea of using the Lagrange multiplier as a price in the control of a multi-agent system is quite common in power systems control. Such an idea is used in [1] to indirectly control power generators in a multi-agent system. However, such a control scheme relies on a hierarchal communication scheme and is therefore inherently not decentralized. The idea of a price controller is also explored by [15], which develops a KKT based controller for the steady state control of a system. This idea is similar to the material presented in this section in that the controller solves both a price-based prob-

lem and the dynamic optimization problem simultaneously. Furthermore, the problem formulation presented in the paper allows for constraints on the states. Nevertheless, the controller only guarantees optimal states in steady state, while making no guarantees for any characteristics of the state before steady state. In the presence of ever changing demand, whether the states of our problem reach steady state is unclear. Our algorithm is applicable regardless of whether the states of the system reach steady state.

We begin the development of our algorithm in the next section and show that it solves the LQR optimization problem for systems with decouple dynamics. Then, we examine the LQR optimization problem with coupled dynamics. With a change in the communication scheme in the multi-agent system, we show that our algorithm applies directly to systems with coupled dynamics as well. Finally, we provide inspiration for future control algorithm development based on market trading by examining a non-Tatonnement price converging model. This chapter assumes the knowledge of convex optimization. For a general overview, please see Appendix A.

4.2 Tatonnement-based Algorithm for Decoupled Dynamic System

To begin the development of our Tatonnement-based algorithm, we first apply dual composition to the LQR problem with decoupled dynamics as presented in Chapter 2. Recall from the optimization problem (2.2) and constraints (2.7)-(2.9) that

$$\begin{aligned} J(x_1 \ldots x_q) &= \min_{u_1,u_2} \sum_{i=1}^{q} \sum_{k=1}^{N-1} (x_i(k)' Q_i x_i(k) + u_i(k)' R_i u_i(k)) + x_i(N)' Q_i x_i(N) \\ \text{s.t.} \quad x_i(k+1) &= A_i x_i(k) + B_i u_i(k) \\ y_i(k) &= C_i x_i(k) \\ \sum_{i=1}^{q} y_i(k) &= P(k), \quad \forall i \in I \end{aligned} \tag{4.1}$$

For simplicity of notation, let us introduce the following definitions.

$$\bar{x}_i(N) = \begin{bmatrix} x_i(1) \\ x_i(2) \\ \vdots \\ x_i(N) \end{bmatrix}, \quad \bar{u}_i(N-1) = \begin{bmatrix} u_i(1) \\ u_i(2) \\ \vdots \\ u_i(N-1) \end{bmatrix}, \quad \bar{C}_i = \begin{bmatrix} C_i & 0 & \cdots & 0 \\ 0 & C_i & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & C_i \end{bmatrix},$$

$$\bar{P}(N) = \begin{bmatrix} P(1) \\ P(2) \\ \vdots \\ P(N) \end{bmatrix}, \quad E_{1i} = \begin{bmatrix} I \\ A_i \\ A_i^2 \\ \vdots \\ A_i^{(N-1)} \end{bmatrix}, \quad E_{2i} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ B_i & 0 & \cdots & 0 \\ A_i B_i & B_i & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A_i^{N-2} B_i & A_i^{N-3} B_i & \cdots & B_i \end{bmatrix}.$$

From (4.1), we can derive the relationship between $\bar{x}_i(N)$ and $\bar{u}_i(N-1)$ as

$$\bar{x}_i(N) = E_{1i}x_i(1) + E_{2i}\bar{u}_i(N-1), \quad \forall i \in I. \tag{4.2}$$

4.2.1 Optimal Control through Dual Decomposition

We can put the optimization problem into its dual form by introducing a Lagrange multiplier vector $\lambda(k) \in \mathbb{R}^m$ [5]. We also define $L(x_1...x_q, u_1...u_q, \lambda)$ as

$$L(x_1...x_q, u_1...u_q, \lambda) = \sum_{i=1}^{q}\sum_{k=1}^{N-1}\left(x_i(k)'Q_ix_i(k) + u_i'(k)R_iu_i(k)\right) + x_i(N)'Q_ix_i(N) + \sum_{k=1}^{N}\lambda(k)'\left(\sum_{i=1}^{q}y_i(k)\right).$$

Then the dual problem becomes

$$\bar{D}(x_1...x_q, \lambda) = \min_{u_1,...,u_q} L(x_1...x_q, u_1...u_q, \lambda) \tag{4.3}$$

$$\hat{D}(x_1...x_q, \lambda) = \bar{D}(x_1...x_q, \lambda) - \sum_{k=1}^{N}\lambda(k)'P(k) \tag{4.4}$$

$$D(x_1...x_q) = \max_{\lambda}\hat{D}(x_1...x_q, \lambda) \tag{4.5}$$

where $\hat{D}(x_1...x_q, \lambda)$ is the dual function.

From convex duality theory, the dual function is a lower bound of the optimal cost J^* in (4.1). The optimal value D^* for (4.5) is the best upper bound of J^*. Since the structure of (4.1) is convex, the control input u_i lives in an open set, and we only have an equality constraint, we have strong duality by the Slater's Condition and consequently $D^* = J^*$ [5]. Therefore, we can find the minimum value of (4.1) by finding D^* distributively. From the structure of (4.4), we can apply dual decomposition to (4.3) and introduce a Tatonnement-based algorithm to completely decompose the dual problem. Since the variables of (4.3) are local and separable for a fixed λ, we can apply dual decomposition to decompose (4.3) into

$$\bar{D}(x_1...x_q, \lambda) = \sum_{i=1}^{q}\bar{D}_i(x_i, \lambda), \tag{4.6}$$

$$\tag{4.7}$$

where $\bar{D}_i$ is defined as

$$\bar{D}_i(x_i, \lambda) = \min_{u_i}\sum_{k=1}^{N-1}\left(x_i(k)'Q_ix_i(k) + u_i(k)'R_iu_i(k) + \lambda(k)'C_ix_i(k)\right) + x_i(N)'Q_ix_i(N) + \lambda(N)'C_ix_i(N), \tag{4.8}$$

to find the optimal control [4, 29]. Note that each agent uses the same $\lambda(k)$ value. The advantage of this decomposition is that the minimization of the partial Lagrangian can

be performed by individual agents in parallel once λ is fixed. Therefore, we can obtain an expression of the optimal control $u_{i_{opt}}$ for each agent as a function of λ.

Let us consider the ith agent in the system. We will derive the optimal control over the entire N time horizon by defining

$$\bar{\lambda}(N) = \begin{bmatrix} \lambda(1) \\ \lambda(2) \\ \vdots \\ \lambda(N) \end{bmatrix}, \quad \bar{Q}_i = \begin{bmatrix} Q_i & 0 & \cdots & 0 \\ 0 & Q_i & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Q_i \end{bmatrix}, \quad \bar{R}_i = \begin{bmatrix} R_i & 0 & \cdots & 0 \\ 0 & R_i & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_i \end{bmatrix}.$$

Then, we can rewrite (4.8) as

$$\begin{aligned} \bar{D}_i(\bar{x}_i(N), \bar{\lambda}(N)) = \min_{\bar{u}_i(N-1)} \; & \bar{x}_i(N)' \bar{Q}_i \bar{x}_i(N) + \bar{u}_i(N-1)' \bar{R}_i \bar{u}_i(N-1) \\ & + \bar{\lambda}(N)' \bar{C}_i \bar{x}_i(N). \end{aligned} \tag{4.9}$$

Recalling the relationship between $\bar{x}_i(N)$ and $\bar{u}_i(N-1)$ from (4.2), we can further rewrite (4.9) as

$$\begin{aligned} \bar{D}_i(\bar{\lambda}(N)) = \min_{\bar{u}_i(N-1)} \; & M_{Ai} + \bar{\lambda}(N)' M_{Bi} + \bar{u}_i(N-1)' \left(2M_{Ci} + M_{Di} \bar{\lambda}(N)\right) \\ & + \bar{u}_i(N-1)' M_{Ei} \bar{u}_i(N-1), \end{aligned} \tag{4.10}$$

where

$$\begin{aligned} & M_{Ai} = x_i(1)' E_{1i}' \bar{Q}_i E_{1i} x_i(1), \quad M_{Bi} = \bar{C}_i E_{1i} x_i(1), \quad M_{Ci} = E_{2i}' \bar{Q}_i E_{1i} x_i(1), \\ & M_{Di} = E_{2i}' \bar{C}_i', \quad M_{Ei} = E_{2i}' \bar{Q}_i E_{2i} + \bar{R}_i. \end{aligned}$$

The function $\bar{D}_i(\bar{\lambda}(N))$ now depends only on $\bar{\lambda}(N)$. We know that M_{Ei} is a symmetric matrix because $\bar{Q}_i$ and $\bar{R}_i$ are symmetric by construction. Because we have established that our problem has strong duality, by KKT conditions (A.5), we can find the optimal control by setting

$$\frac{\partial \bar{D}_i}{\partial \bar{u}_i} = \left(2M_{Ci} + M_{Di} \bar{\lambda}(N)\right) + 2M_{Ei} \bar{u}_i(N-1) = 0.$$

Because $M_{Ei} \succ 0$ from the definitions of $\bar{R}_i$ and $\bar{Q}_i$, M_{Ei} is invertible and the optimal control exists. Therefore, we can find an optimal control $\bar{u}_{i_{opt}}(N-1)$ as a function of $\bar{\lambda}(N)$ with

$$\bar{u}_{i_{opt}}(N-1) = -M_{Ei}^{-1} \left(M_{Ci} + \frac{1}{2} M_{Di} \bar{\lambda}(N) \right). \tag{4.11}$$

We have found a way to distributively compute the optimal control as a function of $\bar{\lambda}(N)$ for (4.10). Next, we want to find a way for each agent to find the optimal $\bar{\lambda}(N)$ that maximizes (4.4) based on this optimal control. We recognize that we cannot

decompose (4.4), since $\lambda(k)'P(k)$ is a coupling term. However, we introduce a Tatonnement based algorithm for this decoupled dynamics LQR problem that enables each agent to calculate the optimal $\bar{\lambda}(N)$ in parallel and thereby completely decomposes the dual problem.

4.2.2 Tatonnement-based Algorithm

To view the process of finding the optimal $\bar{\lambda}(N)$ as a Tatonnement process, let us consider $\bar{\lambda}(N)$ as the price to be adjusted and the state $\bar{x}_i(N)$ as the demand of the ith agent in the system. Besides the time index k, we introduce an iteration horizon with an index l such that $l \in \{1,2,\cdots,T\}$, where T is the total number of iterations to update $\bar{\lambda}(N)$ until equilibrium occurs. We also assume that each agent i calculates its own $\lambda(k)$. The value of $\lambda(k)$ calculated by agent i at iteration l is denoted by $\lambda^i(k,l)$.

We introduce an equation to update $\bar{\lambda}(N)$ at each iteration l for each agent i by first introducing the following notation

$$\bar{\lambda}^i(N,l) = \begin{bmatrix} \lambda^i(1,l)' & \cdots & \lambda^i(N,l)' \end{bmatrix}',$$

$$\bar{u}_{i_{opt}}(N-1,l) = -M_{Ei}^{-1}\left(M_{Ci} + \frac{1}{2}M_{Di}\bar{\lambda}^i(N,l)\right), \tag{4.12}$$

$$\bar{x}_{i_{opt}}(N,l) = E_{1i}x_i(1) + E_{2i}\bar{u}_{i_{opt}}(N-1,l). \tag{4.13}$$

Note that from the problem formulation, each $\bar{\lambda}^i(N,l)$ is identical for all i's at every iteration l. Then, we introduce update equation for $\bar{\lambda}^i(N,l)$ as

$$\bar{\lambda}^i(N,l+1) = \bar{\lambda}^i(N,l) + \gamma\sum_{i=1}^{q}\bar{C}_i\bar{x}_{i_{opt}}(N,l) - \gamma\bar{P}(N), \tag{4.14}$$

where γ is a particular fixed step size. We claim for the purpose of discussion that (4.14) converges after some iteration T to the solution for (4.5). It follows that the equilibrium value of $\bar{\lambda}^i(N,l)$ becomes the optimal Lagrange multiplier that maximizes (4.4). We will prove this claim in the sequel.

To interpret the proposed algorithm in the Economics setting, each agent in the system is both the auctioneer and the auctionee in the Tatonnement process. During the time horizon of each iteration, the ith agent plays the role of the auctionee by generating his demand $x_{i_{opt}}(k,l)$ based on the corresponding calculated price $\lambda^i(k,l)$ at every time instant k. After the time horizon reaches N, the agent bundles his demands into $\bar{x}_{i_{opt}}(k,l)$ and the corresponding prices into $\bar{\lambda}^i(N,l)$. Then, all agents in the system exchange their bundled demands with each other. After the exchange, the ith agent becomes the auctioneer and generates the price $\bar{\lambda}^i(N,l+1)$ based on everyone's demands in the system. The iteration increments and the process repeats until T iterations is reached, after which the price is assumed to have converged to the optimal $\bar{\lambda}(N,l)$. This process is formalized in Algorithm 1.

The advantage of this Tatonnement-based process is that each agent can now perform the maximization of the dual function in parallel. Because the optimal response

Algorithm 1 The Tatonnement-Based Algorithm for Decoupled Dynamic System

for all $i \in I$ **do**
  guess $\lambda(1,1), \lambda(2,1) \ldots \lambda(N,1)$ and bundles it to $\bar{\lambda}(N,1)$
  set $\bar{\lambda}^i(N,1) = \bar{\lambda}(N,1) \quad \forall i \in I$
  for $l = 1$ to T **do**
    set $x_i(1,l)$ satisfying $\sum_{i=1}^{q} C_i x_{i_{opt}}(1,l) = P(1)$
    for $k = 1$ to $N-1$ **do**
      find $u_{i_{opt}}(k,l)$ from $\lambda^i(k,l)$ using (4.12)
      update $x_{i_{opt}}(k+1,l)$ using (4.13)
    end for
    bundle $x_i(k,l)$ for all k into $\bar{x}_{i_{opt}}(N,l)$
    bundle $\lambda^i(k,l)$ for all k into $\bar{\lambda}^i(N,l)$
    send $\bar{x}_{i_{opt}}(N,l)$ to jth agent $\forall j \in I, j \neq i$
    receive $\bar{x}_{j_{opt}}(N,l)$ from jth agent $\forall j \in I, j \neq i$
    update $\bar{\lambda}^i(N,l+1)$ using (4.14)
  end for
end for

(4.12) to $\bar{\lambda}^i(N,l)$ is also computed in parallel, we say that our algorithm is a form of decentralize algorithm. However, we note that this algorithm requires each agent to communicate with all other agents in the system.

In the algorithm presented, both the exchange of the bundled optimal states $\bar{x}_{i_{opt}}(N,l)$ and the update of the bundled price $\bar{\lambda}^i(N,l)$ occur after the time horizon ends. An alternative algorithm may be for the exchange of optimal states and update of the price to occur at every time instant k within the time horizon. In the problem formulation in Chapter 2, we assume that the communication between agents in the system is ideal. With this assumption, the original algorithm is essentially the same as the alternative algorithm. If we consider asynchronous communication, then there are significant differences between the two algorithms. In the alternative algorithm, each agent exchanges states and updates the price at his own pace. Therefore, a situation may occur when the agent updates the price with a delayed optimal state from another agent in the system at a particular time instant k. For example, for a time k, an agent i delays its price update. Then, all agents $j \neq i \in I$ updates their price by

$$\check{x}_{i_{opt}}(N,l) = \begin{bmatrix} x_{i_{opt}}(1,l)' & \cdots & x_{i_{opt}}(k,l-1)' & x_{i_{opt}}(k+1,l)' & \cdots & x_{i_{opt}}(N,l)' \end{bmatrix}',$$

$$\bar{\lambda}^j(N,l+1) = \bar{\lambda}^j(N,l) + \gamma \sum_{m \neq i} \bar{C}_m \bar{x}_{m_{opt}}(N,l) + \bar{C}_i \check{x}_{i_{opt}}(N,l) - \gamma \bar{P}(N).$$

In the original algorithm, each update of the price is bundled and occurs after the time horizon. The price is updated for the entire time horizon at the same time. Since the exchange of the states is also bundled, a delay in sending the optimal state from another agent will affect the price of receiving agent for all times k. This means that if an agent i delays its price update at time k,all agents $j \neq i \in I$ updates their price by

$$\bar{\lambda}^j(N,l+1) = \bar{\lambda}^j(N,l) + \gamma \sum_{m \neq i} \bar{C}_m \bar{x}_{m_{opt}}(N,l) + \bar{C}_i \bar{x}_{i_{opt}}(N,l-1) - \gamma \bar{P}(N).$$

The advantages and disadvantages of both algorithms will be investigated in future works.

4.2.3 Convergence of Algorithm for Decoupled Dynamic Systems

We will prove the convergence of Algorithm 1 by proving that the update equation (4.14) converges to a solution for (4.5). This proof follows from the convergence analysis of the gradient descent approach in [33]. A similar proof of the convergence of an algorithm to optimize flow control can be found in [19].

Because in the ideal communication case, each agent's price $\bar{\lambda}^i(N,l)$ at iteration l is identical, for simplicity we let $\bar{\lambda}(N,l) = \bar{\lambda}^i(N,l)$ for all i at iteration l. Recall from (4.2) and (4.10) that, for the ith agent,

$$\begin{aligned}\bar{D}_i(\bar{\lambda}(N,l)) &= M_{Ai} + \bar{\lambda}(N,l)'M_{Bi} + \bar{u}_{i_{opt}}(N-1,l)'\left(2M_{Ci} + M_{Di}\bar{\lambda}(N,l)\right) \\ &\quad + \bar{u}_{i_{opt}}(N-1,l)'M_{Ei}\bar{u}_{i_{opt}}(N-1,l) \\ &= M_{Ai} - M_{Ci}'M_{Ei}^{-1}M_{Ci} + \bar{\lambda}(N,l)'\left(M_{Bi} - M_{Di}'M_{Ei}^{-1}M_{Ci}\right) \\ &\quad - \frac{1}{4}\bar{\lambda}(N,l)'M_{Di}'M_{Ei}^{-1}M_{Di}\bar{\lambda}(N,l).\end{aligned}$$

From (4.6), we see that (4.4) can be rewritten as

$$\hat{D}(\bar{\lambda}(N,l)) = \sum_{i=1}^{q}\bar{D}_i(\bar{\lambda}(N,l)) - \bar{\lambda}(N,l)'\bar{P}(N). \tag{4.15}$$

Because $\bar{D}_i$ is a quadratic concave function in $\bar{\lambda}(N,l)$, equation (4.15) is also a quadratic concave function in $\bar{\lambda}(N,l)$. We define the gradient of $\hat{D}(\bar{\lambda}(N,l))$ with respect to $\bar{\lambda}(N,l)$ as $\nabla\hat{D}(\bar{\lambda}(N,l))$. Then, we can express $\nabla\hat{D}(\bar{\lambda}(N,l))$ as

$$\nabla\hat{D}(\bar{\lambda}(N,l)) = \sum_{i=1}^{q} M_{Bi} - M_{Di}'M_{Ei}^{-1}M_{Ci} - \frac{1}{2}M_{Di}'M_{Ei}^{-1}M_{Di}\bar{\lambda}(N,l) - \bar{P}(N). \tag{4.16}$$

Using definitions of M_{Bi}, M_{Ci} and equation (4.2), we derive from equation (4.16) the relationship

$$\begin{aligned}\nabla\hat{D}(\bar{\lambda}(N,l)) &= \sum_{i=1}^{q}\bar{C}_iE_{1i}x_i(1) - \bar{C}_iE_{2i}M_{Ei}^{-1}(M_{Ci} - \frac{1}{2}M_{Di}\bar{\lambda}(N,l)) - \bar{P}(N) \\ &= \sum_{i=1}^{q}\bar{C}_i(E_{1i}x_i(1) + E_{2i}\bar{u}_{i_{opt}}(N-1,l)) - \bar{P}(N) \\ &= \sum_{i=1}^{q}\bar{C}_i\bar{x}_{i_{opt}}(N,l) - \bar{P}(N).\end{aligned} \tag{4.17}$$

Then, we can rewrite the update equation (4.14) for $\bar{\lambda}(N,l)$ as

$$\bar{\lambda}(N,l+1) = \bar{\lambda}(N,l) + \gamma\nabla\hat{D}(\bar{\lambda}(N,l)). \tag{4.18}$$

We are now ready to prove the convergence of (4.18) to the optimal $\bar{\lambda}(N)^*$ that solves

(4.5). This is done first by showing that $\nabla\hat{D}(\bar{\lambda}(N,l))$ is Lipschitz continuous. Then we show that for a small enough γ, the limit of the gradient of $\hat{D}(\bar{\lambda}(N,l))$ is zero, which implies (4.18) indeed does converge. Furthermore, since $\hat{D}$ is a quadratic concave function and the gradient $\nabla\hat{D}(\bar{\lambda}(N,l))$ is affine, we see that (4.18) converges to the optimal $\bar{\lambda}(N,l)^*$. because $\nabla\hat{D}(\bar{\lambda}(N,l)^*) = 0$ satisfies the KKT conditions for the maximization of the function $\hat{D}$. Our proof is based on Proposition 2.1 and 2.3 in [33]. Let us first present a lemma based on [33] that will help us in our proof.

Lemma 2. *If a function $\nabla D : \mathbb{R}^n \to \mathbb{R}$ is Lipschitz continuous, i.e. $\exists\, K > 0$ such that*

$$||\nabla D(x) - \nabla D(y)||_2 \leq K\,||x-y||_2\,, \quad \forall x,y \in \mathbb{R}^n,$$

then it follows that

$$D(x-y) \geq D(x) - y^T\nabla D(x) - \frac{1}{2}K||y||_2^2$$

for all $x,y \in \mathbb{R}^n$.

This Lemma is a concave version of Proposition A.32 in Appendix A of [33]. It's proof is straightforward from Proposition A.32. Next, we state our main theorem, which is based on Proposition 2.1 in [33].

Theorem 2. *Suppose that $\bar{\lambda}(N,l)$ is updated according to* (4.18)*. If we choose $0 < \gamma < \frac{2}{K}$, where K is a Lipschitz constant for $\hat{D}$, then*

$$\lim_{l\to\infty}\nabla\hat{D}(\bar{\lambda}(N,l)) = 0 \text{ and } \lim_{l\to\infty}\bar{\lambda}(N,l) = \bar{\lambda}(N)^*,$$

where $\bar{\lambda}(N)^*$ is the solution to (4.5)*.*

Proof. Let us first show that $\nabla\hat{D}$ is Lipschitz continuous. The condition for $\nabla\hat{D}$ to be Lipschitz continuous is that there exists a $K \in \mathbb{R}$ such that

$$\Delta GD(\bar{\lambda}_1,\bar{\lambda}_2) = \left|\left|\nabla\hat{D}(\bar{\lambda}_1(N,l)) - \nabla\hat{D}(\bar{\lambda}_2(N,l)\right|\right|_2 \leq K\left|\left|\bar{\lambda}_1(N,l) - \bar{\lambda}_2(N,l)\right|\right|_2.$$

Since $\nabla\hat{D}(\bar{\lambda}(N,l))$ is an affine function, it is continuously differentiable. Let us define $N_i = \frac{1}{2}M'_{Di}M_{Ei}^{-1}M_{Di}$ and choose $K = \left|\left|\sum_{i=1}^q N_i\right|\right|_2 + \varepsilon$, for some $\varepsilon \in \mathbb{R}$ such that $0 < \varepsilon < \infty$. From (4.16), we have

$$\begin{aligned}\Delta GD(\bar{\lambda}_1,\bar{\lambda}_2) = \left|\left|\sum_{i=1}^q N_i(\bar{\lambda}_1(N,l) - \bar{\lambda}_2(N,l))\right|\right|_2 &\leq \left|\left|\sum_{i=1}^q N_i\right|\right|_2 \left|\left|\bar{\lambda}_1(N,l) - \bar{\lambda}_2(N,l)\right|\right|_2 \\ &\leq K\left|\left|\bar{\lambda}_1(N,l) - \bar{\lambda}_2(N,l)\right|\right|_2\end{aligned}$$

Because $\bar{Q}_i \succ 0$ and $\bar{R}_i \succeq 0$ by definition, we know that $M_{Ei} \succ 0$, and hence M_{Ei}^{-1} exists. The matrix M_{Di} is finite since it is dependent on system matrices A_i, B_i, and C_i. Therefore, $K < \infty$, so $K \in \mathbb{R}^+$. It follows that $\nabla\hat{D}(\bar{\lambda}(N,l))$ is Lipschitz continuous.

Then, from Lemma 2 and the definition of (4.14), we obtain

$$\begin{aligned}\hat{D}(\bar{\lambda}(N,l+1)) &= \hat{D}(\bar{\lambda}(N,l) - (-\gamma\nabla\hat{D}(\bar{\lambda}(N,l))))\\ &\geq \hat{D}(\bar{\lambda}(N,l)) + \gamma\nabla\hat{D}(\bar{\lambda}(N,l))(\nabla\hat{D}(\bar{\lambda}(N,l))) - \frac{1}{2}K||-\gamma\nabla\hat{D}(\bar{\lambda}(N,l))||_2^2\\ &= \hat{D}(\bar{\lambda}(N,l)) + \gamma\left|\left|\nabla\hat{D}(N,l)\right|\right|_2^2 - \frac{1}{2}K\gamma^2\left|\left|\nabla\hat{D}(N,l)\right|\right|_2^2\\ &= \hat{D}(\bar{\lambda}(N,l)) + \gamma\left(1-\frac{K\gamma}{2}\right)\left|\left|\nabla\hat{D}(N,l)\right|\right|_2^2\end{aligned}$$

Let $\beta = \gamma\left(1-\frac{K\gamma}{2}\right)$. Notice that $\beta > 0$ since we chose $0 < \gamma < \frac{2}{K}$. From the construction of the problem, $\hat{D}(\bar{\lambda}(N,l))$ is bounded above by the optimal value J^* of the original optimization problem by duality [5]. This implies that there exits a $M \in \mathbb{R}$ such that $\hat{D}(\bar{\lambda}(N,l)) < M$ for all l. It follows that

$$\begin{aligned}M \geq \hat{D}(\bar{\lambda}(N,l+1)) \geq \hat{D}(\bar{\lambda}(N,1)) + \beta\sum_{\tau=1}^{l}\left|\left|\nabla\hat{D}(\bar{\lambda}(N,\tau))\right|\right|_2^2\\ \implies \sum_{\tau=1}^{\infty}\left|\left|\nabla\hat{D}(\bar{\lambda}(N,\tau))\right|\right|_2^2 \leq \frac{1}{\beta}(M-\hat{D}(\bar{\lambda}(N,1))) < \infty\end{aligned}$$

It follows that $\lim_{l\to\infty}\nabla\hat{D}(\bar{\lambda}(l)) = 0$. Let us denote the solution to the optimization problem (4.5) as $\bar{\lambda}(N)^*$. From the fact that $M_{Ei}^{-1} \succ 0$, since $M_{Ei} \succ 0$, we know that $\bar{\lambda}(N)^*$ is a unique solution that occurs when $\nabla\hat{D}(\bar{\lambda}(N)^*) = 0$ by KKT conditions. Since $\nabla\hat{D}(\bar{\lambda}(N,l))$ is affine and $\lim_{l\to\infty}\nabla\hat{D}(\bar{\lambda}(N,l)) = 0$, we see that $\lim_{l\to\infty}\bar{\lambda}(N,l) = \bar{\lambda}(N)^*$. Note that From (4.17), we see that (4.14) is in fact a gradient ascend algorithm. □

4.2.4 Simulation

We now provide two simulations that confirms our theory regarding the feasibility of the control and the convergence of the tatonnement algorithm. Let $q = m = n = 2$, $N = 3$, $T = 200$ and consider a system with no feasible controller and the following system and cost characteristics

$$A_1 = \begin{bmatrix} -0.2 & 0 \\ 0 & -0.6 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 2 & 0 \\ 0 & 5 \end{bmatrix}, \quad B_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad B_2 = \begin{bmatrix} -\frac{5}{6} \\ 1 \end{bmatrix},$$
$$C_1 = \begin{bmatrix} 2 & 2 \end{bmatrix}, \quad C_2 = \begin{bmatrix} 1 & \frac{5}{6} \end{bmatrix}, \quad x_1(1,1) = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \quad x_2(1,1) = \begin{bmatrix} 5 \\ 6 \end{bmatrix},$$
$$\lambda(1,1) = \lambda(2,1) = \lambda(3,1) = -50, \quad Q_1 = C_1'C_1, Q_2 = C_2'C_2, R_1 = R_2 = 1.$$

For simplicity, we set $P(k) = 20$ to be constant for all k. From the system characteristics, we calculated the parameter $K \approx 0.6262$, which dictates that if the control algorithm is feasibility, then it will converge to the optimal $\bar{\lambda}^*(N)$ if we pick $0 < \gamma < 3$. Let us pick $\gamma = 0.2$. From construction, $C_1B_1 = 0$ and $C_2B_2 = 0$. By the discussion in the next section, this system has does not have a feasible control, and we expect

$\lambda(k,l)$ to diverge for some value k as iterations increase. Indeed, $\lambda(2)$ diverges almost immediately after the start of the algorithm. This divergence is illustrated in Figure 4.1.

Suppose now we change B_1 and B_2 to the following

$$B_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, B_2 = \begin{bmatrix} 3 \\ 4 \end{bmatrix}.$$

Since C_1B_1 and C_2B_2 are not equal to zero, the $\lambda(k)$ values should converge for all k. From system characteristics, we calculate $K \approx 0.9653$, which dictates that the system will converge provided $0 < \gamma < 2$. Let us pick again $\gamma = 0.2$. The convergence of $\lambda(k)$ is illustrated in Figure 4.2. This convergence occurs around the 38th iteration.

4.2.5 A Discussion on the Feasibility of the Controller

In order for the controller to be feasible, it must satisfy the equality constraint

$$\sum_{i=1}^{q} \bar{C}_i \bar{x}_i(N) = \bar{P}(N). \tag{4.19}$$

Proposition 1. *There exists a feasible controller that satisfies* (4.19) *if and only if* $C_iB_i \in \mathbb{R}^{1\times m}$ *is not a zero vector for all* $i \in I$.

Proof. We note that (4.19) can then be rewritten as

$$\sum_{i=1}^{q} \bar{C}_i E_{2i} \bar{u}_i(N-1) = \bar{P}(N) - \sum_{i=1}^{q} \bar{C}_i E_{1i} x_i(1). \tag{4.20}$$

The initial conditions of the optimization problem dictates that the equality constraint is satisfied at $k = 1$ independent of the control. Therefore, the first row of (4.20) have no effect on the feasibility of control. Therefore, let us define

$$\bar{P}(N) = \begin{bmatrix} P(1) \\ \hat{P}(N) \end{bmatrix}, \quad \bar{C}_i = \begin{bmatrix} C_i & 0 \\ 0 & \hat{C}_i. \end{bmatrix}, \quad E_{1i} = \begin{bmatrix} I \\ \hat{E}_{1i} \end{bmatrix}, \quad E_{2i} = \begin{bmatrix} 0 \\ \hat{E}_{2i} \end{bmatrix}.$$

We consider

$$\sum_{i=1}^{q} \hat{C}_i \hat{E}_{2i} \bar{u}_i(N-1) = \hat{P}(N) - \sum_{i=1}^{q} \hat{C}_i \hat{E}_{1i} x_i(1)$$

Let us define

$$F = \begin{bmatrix} \hat{C}_1\hat{E}_{21} & \hat{C}_2\hat{E}_{22} & \cdots & \hat{C}_q\hat{E}_{2q} \end{bmatrix}$$

$$F \begin{bmatrix} \bar{u}_1(N-1) \\ \vdots \\ \bar{u}_q(N-1) \end{bmatrix} = \hat{P}(N) - \sum_{i=1}^{q} \hat{C}_i \hat{E}_{1i} x_i(1). \tag{4.21}$$

The rank of the matrix F is at most $(N-1)$, which corresponds to the number of

constraints that needs to be satisfied by some control, since we want the controller to be feasible at every time step after the initial time $k = 1$. Furthermore, F is always a $(N-1) \times q(N-1)m$ matrix. Because the feasible control does not exists if and only if the row rank of F is less than $N-1$, in which case the system of linear equations (4.21) has no solution, a feasible control exists if and only if F has rank $N-1$. Let us consider the structure of F. Each block matrix $\hat{C}_i\hat{E}_{2i}$ is in the form

$$\hat{C}_i\hat{E}_{2i} = \begin{bmatrix} C_iB_i & 0 & 0 & \cdots & 0 \\ C_iA_iB_i & C_iB_i & 0 & \cdots & 0 \\ \vdots & \cdots & \ddots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \ddots & \vdots \\ C_iA_i^{N-2}B_i & \cdots & \cdots & \cdots & C_iB_i \end{bmatrix}, \quad C_iB_i \in \mathbb{R}^{1\times m}. \tag{4.22}$$

Because $C_iB_i \in \mathbb{R}^{1\times m}$, from the structure of $\hat{C}_i\hat{E}_{2i}$, we see that the matrix F has rank less than $(N-1)$ if and only if C_iB_i is a zero vector for all $i \in I$. □

Suppose C_iB_i is a zero vector for an agent i. From equation (4.2), we see that is equivalent to $C_i\bar{x}_i(N) = C_iE_1ix_i(1)$, meaning that the controller for the ith agent cannot influence the output of agent i at that time instant. In the application of wind turbines, the condition that the controller is not feasible, i.e. $C_iB_i = 0$ for all $i \in I$, corresponds to none of the controllers of the wind turbine affects the energy generated by the wind turbine at that time instant. Since wind turbines are designed such that the controllers affects the energy generated by the turbine, a feasible controller always exists.

4.3 Tatonnement-Based Algorithm for Coupled Dynamics

We apply the same idea from the previous section to an LQR problem with dynamic coupling. Recall the definitions in (2.3). Also, recall the optimization problem (2.4) and it's constraints (2.5) - (2.6) as

$$\begin{aligned} J(x_1 \ldots x_q) &= \min_u \sum_{i=1}^{N} x(k)'Qx(k) + u(k)'Ru(k)) + x(N)'Qx(N) \\ x(k+1) &= Ax(k) + Bu(k) \\ Cx(k) &= P(k). \end{aligned} \tag{4.23}$$

Let us define

$$\hat{x}(N) = \begin{bmatrix} x(2) \\ \vdots \\ x(N) \end{bmatrix}, \quad \hat{u}(N-1) = \begin{bmatrix} u(1) \\ \vdots \\ u(N-1) \end{bmatrix}, \quad \hat{P}(N) = \begin{bmatrix} P(2) \\ \vdots \\ P(N) \end{bmatrix},$$

$$\hat{C} = \begin{bmatrix} C & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & C \end{bmatrix}, \quad \hat{Q} = \begin{bmatrix} Q & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & Q \end{bmatrix}, \hat{R} = \begin{bmatrix} R & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & R \end{bmatrix},$$

$$\hat{A} = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^{N-1} \end{bmatrix}, \quad \hat{B} = \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-2}B & A^{N-3}B & \cdots & B \end{bmatrix}.$$

Then, we can write the dynamics of (4.23) as an algebraic equation $\hat{x}(N) = \hat{A}x(1) + \hat{B}\hat{u}(N-1)$. Furthermore, we can rewrite (4.23) as

$$\begin{aligned} J(\hat{x}) = \min_{\hat{u}(N-1)} \; & \hat{x}(N)'\hat{Q}\hat{x}(N) + \hat{u}(N-1)'\hat{R}\hat{u}(N-1) + x(1)'Qx(1) \\ \text{s.t.} \quad & \hat{C}\hat{x}(N) = \hat{P}(N). \end{aligned}$$

Combine the cost function with the definition for $\hat{x}(N)$, we have

$$\begin{aligned} J(\hat{x}) = \min_{\hat{u}(N-1)} \; & x(1)'(\hat{A}'\hat{Q}\hat{A} + Q)x(1) + 2x(1)'\hat{A}'\hat{Q}\hat{B}\hat{u}(N-1) + \hat{u}(N-1)'(\hat{B}'\hat{Q}\hat{B} + \hat{R})\hat{u}(N-1) \\ \text{s.t.} \quad & \hat{C}\hat{A}x(1) + \hat{C}\hat{B}\hat{u}(N-1) - \hat{P}(N) = 0. \end{aligned} \tag{4.24}$$

Similar to the previous section, we can define a Lagrangian function as

$$\begin{aligned} L(\hat{u}(N-1), \bar{\lambda}(N)) = {} & x(1)'\left(\hat{A}'\hat{Q}\hat{A} + Q\right)x(1) + 2x(1)'\hat{A}'\hat{Q}\hat{B}\hat{u}(N-1) \\ & + \hat{u}(N-1)'\left(\hat{B}'\hat{Q}\hat{B} + \hat{R}\right)\hat{u}(N-1) + \bar{\lambda}(N)'(\hat{C}\hat{B}\hat{u}(N-1)) \\ & + \bar{\lambda}(N)'(\hat{C}\hat{A}x(1) - \hat{P}(N)), \end{aligned}$$

where $\bar{\lambda}(N)$ is defined the same way as in the previous section. Let us define

$$\begin{aligned} H(\bar{\lambda}(N)) = \min_{\hat{u}(N-1)} \; & x(1)'(\hat{A}'\hat{Q}\hat{A} + Q)x(1) + 2x(1)'\hat{A}'\hat{Q}\hat{B}\hat{u}(N-1) \\ & + \hat{u}(N-1)'(\hat{B}'\hat{Q}\hat{B} + \hat{R})\hat{u}(N-1). \end{aligned} \tag{4.25}$$

From (4.25), we have the necessary condition by KKT

$$2x(1)'\hat{A}'\hat{Q}'\hat{B} + 2\hat{u}(N-1)'(\hat{B}'\hat{Q}\hat{B} + \hat{R}) + \bar{\lambda}(N)'\hat{C}\hat{B} = 0. \tag{4.26}$$

Assumption 3. *The dynamics and the cost function of the entire system, namely matrices $\hat{A}$, $\hat{B}$, $\hat{Q}$, $\hat{R}$, are known for all agents in the system.*

Let $E = (\hat{B}'\hat{Q}\hat{B} + \hat{R})^{-1}$. This matrix is guarantee to exist because $(\hat{B}'\hat{Q}\hat{B} + \hat{R})$ is a positive definite matrix by the definition of $\hat{Q}$ and $\hat{R}$. Let us also define $U_1 = \hat{B}'\hat{C}'$, and $U_2 = \hat{B}'\hat{Q}\hat{A}x(1)$. Then, similar to (4.11), we derive the optimal control from (4.26) as a function of $\bar{\lambda}(N)$:

$$\hat{u}_{opt}(N-1) = -\frac{1}{2}E'U_1\bar{\lambda}(N) - E'U_2.$$

Recalled the notation $\bar{\lambda}^i(N,l)$ denotes the value of $\bar{\lambda}(N)$ calculated by agent i at iteration l of an update equation. Since $\bar{\lambda}^i(N,l)$ is identical for all i at every iteration l for an ideal communication scheme, we let $\bar{\lambda}(N,l) = \bar{\lambda}^i(N,l)$ for all i. Let $\hat{x}_{opt}(N,l) = \hat{A}x(1) + \hat{B}u_{opt}(N-1,l)$ be the optimal response state of the system to $\bar{\lambda}(N,l)$. For the coupled dynamic system, we have the following updating equations for $\bar{\lambda}^i(N,l)$, similar to (4.14),

$$\begin{aligned}\bar{\lambda}^i(N,l+1) &= \bar{\lambda}^i(N,l) + \gamma\hat{C}\hat{x}_{opt}(N,l) - \gamma\hat{P}(N) \\ &= \bar{\lambda}^i(N,l) + \gamma\hat{C}(\hat{A}x(1) + \hat{B}\hat{u}_{opt}(N-1,l)) - \gamma\hat{P}(N). \end{aligned} \tag{4.27}$$

where

$$\hat{u}_{opt}(N-1,l) = -\frac{1}{2}E'U_1\bar{\lambda}(N,l) - E'U_2. \tag{4.28}$$

Then, (4.25) can be rewritten as

$$\begin{aligned}H(\bar{\lambda}(N,l)) &= \hat{u}_{opt}(N-1,l)'E^{-1}\hat{u}_{opt}(N-1,l) + 2U_2'\hat{u}_{opt}(N-1) \\ &\quad + \bar{\lambda}(N,l)'U_1'\hat{u}_{opt}(N-1,l) \\ &= \left(-\frac{1}{2}E'U_1\bar{\lambda}(N,l) - E'U_2\right)' E^{-1}\left(-\frac{1}{2}E'U_1\bar{\lambda}(N,l) - E'U_2\right) \\ &\quad 2U_2'\left(-\frac{1}{2}E'U_1\bar{\lambda}(N,l) - E'U_2\right) + \bar{\lambda}(N,l)'U_1'\left(-\frac{1}{2}E'U_1\bar{\lambda}(N,l) - E'U_2\right) \\ &= -\frac{1}{4}\bar{\lambda}(N,l)'U_1'E'U_1\bar{\lambda}(N,l) - U_2'E'U_1\bar{\lambda}(N,l) - U_2'E'U_2. \end{aligned} \tag{4.29}$$

We define the dual function is defined as

$$\hat{H}(\bar{\lambda}(N)^*) = \max_{\bar{\lambda}(N)} H(\bar{\lambda}(N)) + \bar{\lambda}(N)'(\hat{C}\hat{A}x(1) - \hat{P}(N)). \tag{4.30}$$

From the definition of U_1 and (4.29), we see that

$$\nabla\hat{H}(\bar{\lambda}(N,l)) = \hat{C}(\hat{A}x(1) + \hat{B}\hat{u}_{opt}(N-1,l)) - \hat{P}(N),$$

and therefore, for each agent i, we can rewrite (4.27) as

$$\bar{\lambda}^i(N,l+1) = \bar{\lambda}^i(N,l) + \gamma\nabla\hat{H}(\bar{\lambda}^i(N,l)). \tag{4.31}$$

Notice that this is basically the same algorithm as presented in the previous section.

Using the arguments of the previous section, we can show that this algorithm converges to the optimal $\bar{\lambda}(N)^*$ for the coupled dynamic system.

We claim that $u_{opt}(N-1,l)$ can be computed in parallel by all agents, since there exists a permutation matrix Z such that

$$\begin{bmatrix} \bar{u}_{1_{opt}}(N-1,l) \\ \vdots \\ \bar{u}_{m_{opt}}(N-1,l) \end{bmatrix} = Z\hat{u}_{opt}(N-1,l) = -\frac{1}{2}ZE'U_1\bar{\lambda}(N,l) - ZE'U_2, \tag{4.32}$$

where $\bar{u}_{i_{opt}}(N-1,l)$ is defined in (4.12). By Assumption 3, each agent has access to $E'U_1$ and $E'U_2$. Therefore, there is enough information for each agent i such that the optimal control for the ith agent $\bar{u}_{i_{opt}}$ can be computed in parallel. Furthermore, if each agent communicates his optimal response $\bar{u}_{i_{opt}}$ to all other agents in the system, then using Z^{-1}, each agent can also calculate (4.31) in parallel. Therefore, the algorithm for the coupled system can still be considered as a decentralized algorithm. Recall this is different from the communication scheme of the algorithm for the decoupled dynamic system, where the ith agent needs to share $x_{i_{opt}}$ with every other agent. The algorithm is formalized in Algorithm 2.

Algorithm 2 The Tatonnement-Based Algorithm for Coupled Dynamic Systems

for all $i \in I$ **do**
  guess $\bar{\lambda}(N,1)$ and set $\bar{\lambda}^i(N,1) = \bar{\lambda}(N,1) \quad \forall i \in I$
  for $l = 1$ to T **do**
    find $\bar{u}_{i_{opt}}(N-1,l)$ from $\bar{\lambda}^i(N,l)$ using (4.32)
    send $\bar{u}_{i_{opt}}(N-1,l)$ to jth agent $\forall j \in I, j \neq i$
    receive $\bar{u}_{i_{opt}}(N-1,l)$ from jth agent $\forall j \in I, j \neq i$
    use Z^{-1} to find $\hat{u}_{opt}(N-1,l)$
    find $\bar{\lambda}^i(N,l+1)$ from $\hat{u}_{opt}(N-1,l)$ using (4.27)
  end for
end for

In the ideal communication situation, the main difference in algorithm between the couple dynamic system and the decoupled dynamic system is the information that is transmitted between each agent. Notice that due to the coupled dynamics, this algorithm does not have the choice of calculating $u_{i_{opt}}(k,l)$ implemented at each time step k. Instead, $u_{i_{opt}}(k,l)$ must be calculated for the entire horizon at every iteration l. For the non-ideal communication situation, such as asynchronous communication as discussed at the end of Section 4.2.2, the difference in exchanged information may lead to different convergence behavior of the two algorithms.

As with the algorithm with decoupled dynamics, we see from (4.24) that a feasible control exits if and only if the constraint

$$\hat{C}\hat{B}\hat{u}(N-1) = \hat{P}(N) - \hat{C}\hat{A}x(1)$$

is satisfied. We see from previous discussion that this is possible if and only if $\hat{C}\hat{B}$ has

row rank of $N-1$. The matrix $\hat{C}\hat{B}$ has the structure

$$\hat{C}\hat{B} = \begin{bmatrix} CB & \hat{0} & \cdots & \hat{0} \\ CAB & CB & \cdots & \hat{0} \\ \vdots & \cdots & \ddots & \vdots \\ CA^{N-2}B & \dots & \dots & CB \end{bmatrix}, \quad CB \in \mathbb{R}^{1\times qm}.$$

Therefore, $\hat{C}\hat{B}$ will always have row rank $N-1$ unless CB is a zero vector. When CB is zero, the control for the entire wind farm is not feasible because none of the controls in the wind farm affects the power output of the wind farm, which in reality is never true due to the design of wind turbines.

4.4 Non-Tatonnement Market Model

A drawback of the Tatonnement process is that no trade occurs until the equilibrium price is achieved. However, in applications such as wind farm, the conditions of the environment changes with time. Therefore, the equilibrium price that is achieved by Tatonnement may be the equilibrium for the environment in a previous time, but not in the environment at the present time. In addition, a major challenge for the implementation of the Tatonnement-based algorithm is that the controls generated by the algorithm is not feasible until the convergence of the algorithm is reached. Therefore, if the algorithm cannot be run until completion due to computational limitations or run time requirements, the suboptimal results of the algorithm may not be implementable.

In Economics, the idea of non-Tatonnment trading processes have been proposed to remedy this. Non-tatonnment processes allow trading during price adjustment, when the price has not yet cleared the market. In this process, it is assumed that trading is done in a way such that the number of goods in the market stays constant. This implies that the process is always feasible. With some assumptions on the dynamics of the trading and the price adjustment, it can be proved that such a process still achieves the market clearing price. Although we have not yet adopted the non-Tatonnement process into a control algorithm, we present the idea here as a motivation for future development of price based wind farm controllers. This discussion is based on the work of [11], but we provide a self-contained new proof.

4.4.1 Market Model and Definitions

We consider a market with n agents and m commodities. Mathematically, we define the following scalar values in Table 4.1.

Base on these scalar values, we define the following vectors.

$$x_\alpha(t) = \begin{bmatrix} x_{\alpha 1}(t) \\ \vdots \\ x_{\alpha n}(t) \end{bmatrix}, \quad \bar{x}_\alpha(t) = \begin{bmatrix} \bar{x}_{\alpha 1}(t) \\ \vdots \\ \bar{x}_{\alpha n}(t) \end{bmatrix}, \quad p(t) = \begin{bmatrix} p_1(t) \\ \vdots \\ p_m(t) \end{bmatrix},$$

$$\bar{x}(t) = \begin{bmatrix} \bar{x}_{11}(t) & \cdots & \bar{x}_{n1}(t) \\ \vdots & \ddots & \vdots \\ \bar{x}_{1m}(t) & \cdots & \bar{x}_{nm}(t) \end{bmatrix}, \quad z_\alpha(t) = x_\alpha(t) - \bar{x}_\alpha(t), \quad z(t) = \sum_\alpha z_\alpha(t).$$

Note that $z_\alpha(t)$ can be interpreted as a function $z_\alpha(t)(p(t), \bar{x}_\alpha(t))$.

We define the utility function for agent α as $U_\alpha : \mathbb{R}^m \to \mathbb{R}$ and assume it is a concave function. Furthermore, we assume that the utility function is strictly increasing. This means that for the vectors $x, y \in \mathbb{R}^m$ with $x_i \geq y_i$ for all $i \in \{1, ..., m\}$ and at least one $x_i > y_i$, where x_i and y_i are the ith element of the vectors x and y respectively, we have $U_\alpha(x) > U_\alpha(y)$. We then define the relationship between x_α and U_α as

$$x_\alpha(t) = \arg\max_\omega U_\alpha(\omega(t)) \quad \text{s.t.} \quad p(t)'\omega(t) = p(t)'\bar{x}_\alpha(t).$$

We call the constraint $p(t)'\omega(t) = p(t)'\bar{x}_\alpha(t)$ the budget constraint. Consequently, the excess demand function $z_\alpha(t)$ can defined as

$$z_\alpha(t) = \arg\max_\omega U_\alpha(\omega(t) + \bar{x}_\alpha(t)) \quad \text{s.t.} \quad p(t)'\omega(t) = 0.$$

Next, we model the trading process for agent α as a function

$$\dot{\bar{x}}_\alpha(t) = F_\alpha(p(t), \bar{x}(t)) \tag{4.33}$$

such that $F_\alpha : R^{n\times(m+1)} \to R^n$. We make the following assumptions regarding this trading model.

Assumption 4. *The function F_α has the following properties.*

$$\sum_\alpha F_\alpha(p(t), \bar{x}(t)) = 0 \tag{4.34}$$

$$p(t)'F_\alpha(p(t), \bar{x}(t)) = 0 \tag{4.35}$$

$$z_j(t) z_{\alpha j}(t) \geq 0, \quad \forall t \geq h \quad \forall j, \alpha. \tag{4.36}$$

Assumption (4.34) can be interpreted as a closed market assumption such that the number of commodities within the market is constant throughout the entire process. Assumption (4.35) tells us that the market value at instantaneous price remains constant for all the agents in the market. Finally, last assumption (4.36) tells us that at some time h and for all later times, all agents will trade so as to satisfy inequality (4.36). This assumption is given the name "Hahn process" based on its introduction by Hahn in [11].

Finally, we define the set of equilibrium prices and allocations as

$$E = \{(p^*(t), \bar{x}^*(t)) : p^*(t) \geq 0, z(p^*(t), \bar{x}^*(t)) = 0\}. \tag{4.37}$$

4.4.2 Price Adjustment and Convergence

Our goal is to develop a price adjustment process and show its convergence to a point in the set E. The following process is proposed in [11]:

$$\dot{p}(t) = \begin{cases} 0 & \text{for } t \leq h \\ Kz(t) & \text{otherwise} \end{cases} \tag{4.38}$$

where K is a diagonal matrix whose diagonal elements consist of $k_i > 0 \forall i \in \{1, \cdots, n\}$. We state the following theorem.

Theorem 3. *The point $z(p^*(t), \bar{x}^*(t))$ is a globally asymptotically stable equilibrium of the dynamic system defined by* (4.38) *and* (4.33)*, where F_α satisfies Assumption 4.*

Note that this means for every $(p(t), \bar{x}(t))$, we have $(p(t), \bar{x}(t)) \to (p^*(t), \bar{x}^*(t)) \in E$ as $t \to \infty$.

Proof. For the pair $(p(t), \bar{x}(t))$ to be in the set E, the two conditions that need to be satisfied is that $p(t) \geq 0$ and $z(p(t), \bar{x}(t)) = 0$. We first show that using our price adjustment model, $p(t) \geq 0$ is always true given that the initial condition $p(0) \geq 0$. Since $\dot{p}(t) = 0$ for $t < h$, it is enough to show that $p(t) \geq 0$ for all $t \geq h$. In order for $p(t) \geq 0$ for all $t \geq h$, it is sufficient to prove that the adjustment process (4.38) behaves such that if $p_j(t) = 0$, then $\dot{p}_j(t) \geq 0$.

When $p_j(t) = 0$, we note that the budget constraint is satisfied regardless of the values of $x_{\alpha j}(t)$ and $\bar{x}_{\alpha j}(t)$. However, we know from definition that $x_\alpha(t)$ now becomes

$$x_\alpha(t) = \arg\max_{\omega} U_\alpha(\omega(t)) \quad s.t. \quad p_i(t)'\omega_i(t) = p_i(t)'\bar{x}_{\alpha i}(t) \forall i \neq j.$$

Since the jth element is now basically without constraint, we know from the maximization that $x_{\alpha j}(t)$ will be the demand of all of the available commodities $\bar{x}_j(t)$. Therefore,

$$\begin{aligned} x_{\alpha j}(t) = \bar{x}_j(t) = \sum_{\beta=1}^{\alpha} \bar{x}_{\beta j}(t) \geq \bar{x}_{\alpha j}(t) \\ \implies \dot{p}_j(t) = k_j(x_\alpha(t) - \bar{x}_{\alpha j}(t)) \geq 0 \end{aligned}$$

for $p_j(t) = 0$. Therefore, $p(t) \geq 0$ for all $t \geq h$.

Now, we want to show that $z(p(t), \bar{x}(t)) = 0$ is a stable equilibrium. Let us consider the budget constraint for each agent α and differentiate it such that

$$\begin{aligned} p(t)'z_\alpha(t) = 0 &\implies \dot{p}(t)'z_\alpha(t) + p(t)'(\dot{x}_\alpha(t) - \dot{\bar{x}}_\alpha(t)) = 0 \\ &\implies \dot{p}(t)'z_\alpha(t) - p(t)'\dot{\bar{x}}_\alpha(t) + p(t)'\dot{x}_\alpha(t) = 0. \end{aligned} \tag{4.39}$$

From assumption (4.35), we know that $p(t)'\dot{\bar{x}}_\alpha(t) = 0$. From our definition of the adjustment process, for $t \geq h$, we have $\dot{p}(t)'z_\alpha(t) = z(t)'Kz_\alpha(t) \geq 0$ by assumption (4.36). Now, suppose $p(t) \geq 0$ and $(p(t),\bar{x}(t)) \notin E$, then $\dot{p}(t) \neq 0$. Furthermore, there exists an agent α such that $z(t)'Kz_\alpha(t) > 0$. This can be seen from the following argument. Suppose we assume that $(p(t),\bar{x}(t)) \notin E$ but $z(t)'Kz_\alpha(t) = 0$ for all agents α. This means that $\sum_j k_j z_j(t) z_{\alpha j}(t) = 0$ for all agents α. This implies

$$\sum_\alpha z(t)'Kz_\alpha(t) = z(t)'K\sum_\alpha z_\alpha(t) = z(t)'Kz(t) = 0.$$

From the definition of K, this can only mean that $z(p(t),\bar{x}(t)) = 0$. However, this contradict the assumption that $(p(t),\bar{x}(t)) \notin E$. Therefore,

$$z(t)'Kz_\alpha(t) \begin{cases} \geq 0 & \forall\alpha \\ > 0 & \text{for at least 1 } \alpha \end{cases} \tag{4.40}$$

From (4.40) and (4.39), we see that $p(t)'\dot{x}_\alpha(t) \leq 0$ for all α and there exists at least 1 α such that $p'\dot{x}_\alpha(t) < 0$.

Now let us take the time derivative of the utility function for agent α. We then have the equality

$$\frac{d}{dt}U_\alpha(x_\alpha(t)) = \frac{\partial}{\partial x_\alpha}U_\alpha(x_\alpha(t))\dot{x}_\alpha(t).$$

From the definition of $x_\alpha(t)$ and the definition of the utility function as a concave function, we know that $x_\alpha(t)$ necessarily satisfy the KKT conditions (A.5) such that

$$\frac{\partial U_\alpha}{\partial x_\alpha}(x_\alpha(t)) - \lambda_\alpha p(t)' = 0$$
$$p(t)'x_\alpha(t) = p(t)'\bar{x}_\alpha(t) \quad \forall\alpha.$$

Since U_α is an increasing function, we know that $\lambda_\alpha \geq 0$ for all α. Therefore, assuming that $(p(t),\bar{x}(t)) \notin E$, we have

$$\frac{\partial U_\alpha}{\partial x_\alpha}(x_\alpha(t)) = \lambda_\alpha p(t)' \implies \frac{d}{dt}U_\alpha(x_\alpha(t)) = \begin{cases} \leq 0 & \forall\alpha \\ < 0 & \text{for at least 1 } \alpha \end{cases}$$

Finally, we are going to use the Second Method of Lyapunov to prove stability. Let us define the Lyapunov function $V(t) = \sum_\alpha U_\alpha(x_\alpha(t))$. From the previous discussion, we see that

$$\frac{d}{dt}V(t) \begin{cases} < 0 & \forall(p(t),\bar{x}(t)) \notin E \\ = 0 & \forall(p(t),\bar{x}(t)) \in E \end{cases}$$

Since $V(t) \geq 0$ from the fact that $U_\alpha(x_\alpha(t)) \geq 0$ for all α by definition, $V(t)$ is a valid Lyapunov function. Therefore, by the Second Method of Lyapunov, the equilibrium pair $(p^*(t),\bar{x}^*(t))$ is stable. □

4.5 Tables and Figures

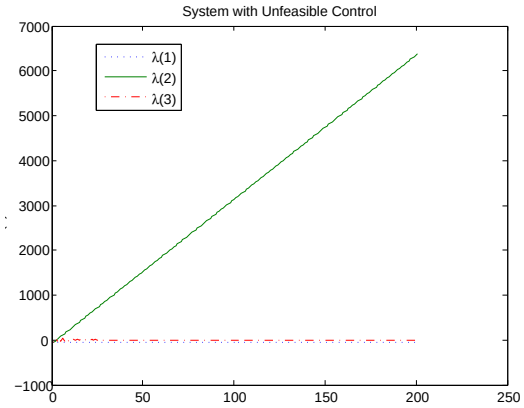

Figure 4.1: The evolution of $\lambda(k,l)$ as iteration increases with $\gamma = 0.2$ and C_1B_1 and C_2B_2 both equal to zero. The $\lambda(2)$ diverges for all l.

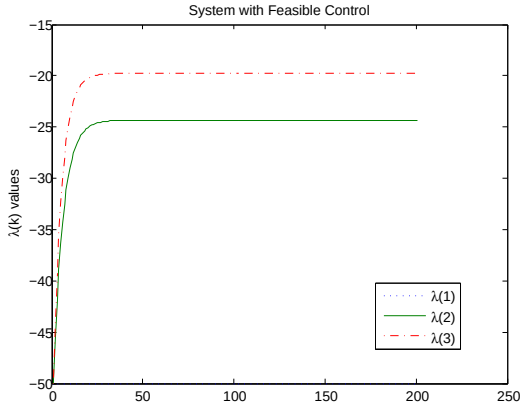

Figure 4.2: The evolution of $\lambda(k,l)$ as iteration increases with $\gamma = 0.2$ and C_1B_1 and C_2B_2 both not equal to zero. All $\lambda(k)$ converges to some value.

Table 4.1: Definitions of scalar variables used in the discussion at time t.

Scalar Variables	Definitions
$x_{\alpha j}(t)$	Gross demand of commodity j for agent α
$\bar{x}_{\alpha j}(t)$	Stock of commodity j for agent α
$z_{\alpha j}(t) = x_{\alpha j}(t) - \bar{x}_{\alpha j}(t)$	Excess demand of commodity j for agent α
$z_j(t) = \sum_\alpha z_{\alpha j}(t)$	Aggregated demand for commodity j
$p_j(t)$	Price for commodity j

Chapter 5

Conclusions and Future Works

5.1 Conclusions

Our goal is to minimize the cost of energy generation from a wind farm in order to maximize the wind farm efficiency. To accomplish our goal, we introduce a two level control scheme that gave each wind turbine the ability to generate its own operating points and optimize its own control, all the while considering the wind farm characteristics. The first level of control is formulated as a stochastic resource allocation problem, while the second level is formulated as a deterministic LQR optimization problem. We impose an equality constraint on both problems to represent the balance between the generated energy and the energy demand.

We have developed two different control algorithms to solve each optimization problem in the two levels of control. The stochastic control algorithm, inspired by a deterministic counterpart, solves the stochastic resource allocation problem almost surely by a quasi-martingale convergence and stochastic gradient descent argument. This algorithm has the added characteristic of being a feasible algorithm. The Tatonnement-based algorithm solves the LQR optimization problem through simultaneously finding the optimal control and the optimal Lagrange multiplier. This algorithm can be applied to both coupled and decoupled dynamic systems. Finally, and most importantly, both algorithms can be implemented distributively, which greatly reduces the computational load on the system.

5.2 Future Work

Each of the proposed algorithms focuses on different aspects of the minimization problem. The Tatonnement-based algorithm focuses on the dynamic aspect of minimizing the cost of generation, while the stochastic control algorithm focuses on the stochastic nature of wind in generating the operating points. However, for the algorithms to be practical in controlling wind farms, there are several issues that need to be solved through future research.

One challenge that the Tatonnement-based algorithm faces is the requirement that each turbine be able to communicate with all other wind turbines in the wind farm. Furthermore, as the algorithm is presented in this thesis, the communication must also be synchronous. However, because of the size of wind farms, the requirement for

synchronous communication of all wind turbines is not practical. To meet this challenge, we will investigate asynchronous incremental subgradient methods such as those in [23]. Furthermore, because each of the wind turbines needs everyone's information, we will also investigate gossip algorithms such as those in [28].

Another challenge that the Tatonnement-based algorithm faces is the feasibility issue. As mentioned in Chapter 3, it is important for the algorithm to be feasible because in reality, we may not be able to run the algorithm until completion. However, since the Tatonnement-based algorithm is not feasible, the suboptimal solution that the algorithm returns before convergence would be useless. Unfortunately, since this algorithm is based on the dual problem, it is inherently infeasible until we have reached the optimal state. However, the non-Tatonnement algorithm, introduced in Chapter 4, may provide the answer. A characteristic of the non-Tatonnment algorithm is an equality constraint that is known as the closed market assumption. This guarantees that during the price adjustment process, this equality constraint is always satisfied. Therefore, a future research direction is to use the same process as the one in Chapter 4 to develop a price mechanism based on the non-Tatonnement process.

One challenge faced by the stochastic control algorithm is the convergence rate. From experiments, the convergence rate of the algorithm depends on the step size. The specific step sizes used in the simulation was determined by trial and error to obtain the best accuracy while achieving a relatively fast convergence rate. Also, the step size affects the communication noise since we multiply the communication noise by the step size in the algorithm. Therefore, to effectively use this algorithm in a wind farm control scheme, we need to develop a rule for the step size so we can optimize the convergent rate of the algorithm.

Finally, the proposed control scheme has not been tested numerically by simulations. Although there are many theoretical advantages to our control scheme compared to the traditional control scheme, the improvement of performance needs to be confirmed by a numerical simulation in a wind farm application. Such a simulation will also be valuable in evaluating the practicality of the proposed two level control scheme as well as improvements upon the implementation of such a control scheme.

Appendix A

Overview of Duality and KKT Conditions for Convex Optimization

Convex optimization is a mathematical optimization technique that has been studied for almost a century. The unique characteristic of convex optimization is the exploitation of the convexity property of the function that is being minimized. Convex optimization has a plethora of applications, from automatic control systems and signal processing to communication analysis and economics. In this chapter, we give a brief overview of the concepts of duality and the Karush-Kuhn-Tucker (KKT) conditions for convex optimization. These concepts provide a basis for the derivation of the controllers that will be described in Chapters 3 and 4. The following overview is based largely on [5].

A.1 Duality and KKT Conditions

To discuss duality and KKT conditions, we first define the concepts of a convex set and a convex function.

Definition 1. *The set C is convex if for any $x_1, x_2 \in C$ and any θ with $0 \leq \theta \leq 1$, we have*

$$\theta x_1 + (1-\theta)x_2 \in C.$$

Definition 2. *A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if the domain of f is a convex set and if for all x, y in the domain of f, for θ with $0 \leq \theta \leq 1$, we have*

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y). \tag{A.1}$$

A function f is strictly convex if (A.1) is a strict inequality whenever $x \neq y$ and $0 < \theta < 1$. We say that a function is affine if (A.1) is always an equality. Finally, we say that f is concave if $-f$ is convex, and strictly concave if $-f$ is strictly convex.

We want to solve the following convex optimization problem.

$$\begin{aligned} & \min_{x \in D} \ f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0 \quad i = 1, ..., m \\ & h_i(x) = 0 \quad i = 1, ..., p \end{aligned} \tag{A.2}$$

with $x \in \mathbb{R}^n$. We call $f_0 : \mathbb{R}^n \to \mathbb{R}$ the objective function. We call $f_i(x) \leq 0$ the inequality

constraint with $f_i : \mathbb{R}^n \to \mathbb{R}$, and we call $h_i(x) = 0$ the equality constraint with $h_i : \mathbb{R}^n \to \mathbb{R}$. We define the set D as

$$D = (\cap_{i=0}^{m} \mathbf{dom}\ f_i) \cap \left(\cap_{i=1}^{p} \mathbf{dom}\ h_i\right).$$

where the notation **dom** h denotes the domain of the function h. We assume that the set D is non-empty. Finally, we define the solution to (A.2) as p^*. We say that p^* is primal feasible if and only if $p^* \neq \infty$ because this implies that a feasible solution exists for (A.2).

A.1.1 Duality

We call the optimization problem (A.2) as the primal problem. We now introduce the concept of duality in convex optimization. Let us define the Lagrangian function $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ associated with the primal problem as

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x).$$

with the **dom** $L = D \times \mathbb{R}^m \times \mathbb{R}^p$. We call λ_i the Lagrange multiplier associated with the ith inequality constraint. Similarly, we call ν_i the Lagrange multiplier for the ith equality constraint. Finally, λ and ν are vectors whose ith elements are λ_i and ν_i respectively. We call λ and ν the dual variables or Lagrange multiplier vectors.

We define the dual function for the primal problem (A.2) as $g(\lambda, \nu) = \inf_{x \in D} L(x, \lambda, \nu)$. Notice that regardless of whether the primal problem is convex, the dual problem is always affine in (λ, ν). We define the Lagrange dual optimization problem as

$$\max_{\lambda, \nu} \ g(\lambda, \nu) \tag{A.3}$$

$$\text{s.t. } \lambda \succeq 0 \tag{A.4}$$

We refer to the pair (λ, ν) as dual feasible if $g(\lambda, \nu) > -\infty$. This is true only when $\lambda \succeq 0$ and $(\lambda, \nu) \in$ **dom** g. We say that the solution to the Lagrangian dual problem is (λ^*, ν^*), and $g(\lambda^*, \nu^*) = d^*$. We call $d^* - p^*$ the optimal duality gap, and we say that the primal problem has the property of strong duality if the optimal duality gap is zero.

We can further see the symmetry of the primal and dual optimization problems as follows. First, let us note that if (x, λ, ν) are primal and dual feasible, then

$$\begin{aligned} \max_{\lambda \succeq 0, \nu} L(x, \lambda, \nu) &= f_0(x) + \max_{\lambda \succeq 0, \nu} \{\sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x)\} \\ &= f_0(x), \end{aligned}$$

The last equality is true because $\sum_{i=1}^{p} \nu_i h_i(x) = 0$ since $h_i(x) = 0$ by feasibility, and $\lambda^* = \arg\max_{\lambda \succeq 0} \sum_{i=1}^{m} \lambda_i f_i(x) = 0$ since $f_i(x) \leq 0$ for $i = 1, ..., m$ and λ^* is constrained to be greater than or equal to zero. Therefore, we see from the definition of the primal

problem that

$$p^* = \min_{x \in D} \max_{\lambda \succeq 0, \nu} L(x, \lambda, \nu).$$

By the definition of the dual problem, we also see that

$$d^* = \max_{\lambda \succeq 0, \nu} \min_{x \in D} L(x, \lambda, \nu).$$

Then, if the optimization problem (A.2) has the property of strong duality, the order of the minimization of x and the maximization of (λ, ν) does not matter.

A.1.2 KKT Condition

Suppose that x^*, λ^*, ν^* are primal and dual feasible. By the first order necessary conditions of optimality, if x^* minimizes $L(x, \lambda^*, \nu^*)$, then

$$\nabla f_0(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla f_i(x^*) + \sum_{i=1}^{p} \nu_i^* \nabla h_i(x^*) = 0.$$

This implies that

$$\begin{aligned} f_i(x^*) &\leq 0, \quad i = 1, ..., m \\ h_i(x^*) &= 0, \quad i = 1, ..., p \\ \lambda_i &\geq 0, \quad i = 1, ..., m \\ \lambda_i f_i(x^*) &= 0, \quad i = 1, ..., m \\ \nabla f_0(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla f_i(x^*) + \sum_{i=1}^{p} \nu_i^* \nabla h_i(x^*) &= 0. \end{aligned} \tag{A.5}$$

These conditions are known as the Karush-Kuhn-Tucker(KKT) conditions. If $f_i, i = 0, ..., m$, are convex functions and $h_i, i = 1, ..., p$, are affine functions in our primal problem (A.2), and $\hat{x}, \hat{\lambda}, \hat{\nu}$ satisfy the KKT conditions, then $\hat{x}$ and $(\hat{\lambda}, \hat{\nu})$ are primal and dual optimal and the optimal duality gap is zero [5].

This conclusion be seen by analyzing each of the KKT conditions in (A.5). From the first two conditions, we see that $\hat{x}$ is primal feasible. From the third condition in (A.5), we see that $\hat{\lambda}_i \geq 0$ implies that the Lagrangian $L(x, \hat{\lambda}, \hat{\nu})$ is convex in x. The combination of the convexity of L and the last condition in (A.5) implies that $\hat{x}$ minimizes $L(x, \hat{\lambda}, \hat{\nu})$, which means that $\hat{x}$ is primal optimal. Therefore, we see that

$$g(\hat{\lambda}, \hat{\nu}) = L(\hat{x}, \hat{\lambda}, \hat{\nu}) = f_0(\hat{x}) + \sum_{i=1}^{m} \hat{\lambda}_i f_i(\hat{x}) + \sum_{i=1}^{p} \hat{\nu}_i h_i(\hat{x}) = f_0(\hat{x})$$

where the last equality comes from the second and fourth condition in (A.5). Therefore, $g(\hat{\lambda}, \hat{\nu}) = (f_0(\hat{x})$ implies zero duality gap and $(\hat{\lambda}, \hat{\nu})$ is dual optimal.

Finally, when the primal problem (A.2) has the property of strong duality, then the KKT conditions are both necessary and sufficient conditions for optimality.

References

[1] A.W. Berger and F.C. Schweppe. Real time pricing to assist in load frequency control. *Power Systems, IEEE Transactions on*, 4(3):920–926, 1989.

[2] F. Borrelli and T. Keviczky. Distributed LQR design for identical dynamically decoupled systems. *IEEE Transactions on Automatic Control*, 53(8):1901–1912, 2008.

[3] L. Bottou. On-line learning and stochastic approximations. In *On-line learning in neural networks*, pages 9–42. Cambridge University Press, 1999.

[4] S. Boyd. Lecture notes for EE364B: Convex Optimization II . www.stanford.edu/class/ee364b/lectures.html, 2007.

[5] S.P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge Univ Pr, 2004.

[6] R. Cole and L. Fleischer. Fast-converging tatonnement algorithms for one-time and ongoing market problems. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 315–324. ACM, 2008.

[7] GP Corten, P. Schaak, and ETG Bot. More Power and Less Loads in Wind Farms:Heat and Flux. In *European Wind Energy Conference & Exhibition, London, UK*, 2004.

[8] G.W.E. Council. Global Wind 2009 Report. *Renewable Energy House, Brussels, Belgium*, 2009.

[9] J.C. Culioli and G. Cohen. Decomposition/coordination algorithms in stochastic optimization. *SIAM Journal on Control and Optimization*, 28:1372, 1990.

[10] A. Giridhar and PR Kumar. Computing and communicating functions over sensor networks. *IEEE Journal on Selected Areas in Communications*, 23(4):755–764, 2005.

[11] F.H. Hahn and T. Negishi. A theorem on non-tatonnement stability. *Econometrica*, 30(3):463–469, 1962.

[12] A.D. Hansen, P. Sørensen, F. Iov, and F. Blaabjerg. Centralised power control of wind farm with doubly fed induction generators. *Renewable Energy*, 31(7):935–951, 2006.

[13] B. Johansson. *On distributed optimization in networked systems*. PhD thesis, Kungliga Tekniska hö gskolan, 2008.

[14] K.E. Johnson and N. Thomas. Wind farm control: addressing the aerodynamic interaction among wind turbines. In *Proceedings of the 2009 conference on American Control Conference*, pages 2104–2109. IEEE Press, 2009.

[15] A. Jokic, M. Lazar, and PPJ Van den Bosch. Real-time control of power systems using nodal prices. *International Journal of Electrical Power & Energy Systems*, 31(9):522–530, 2009.

[16] J.M. Jonkman and M.L. Buhl Jr. FAST users guide. *National Renewable Energy Laboratory, Rept. NREL/EL-500–29798, Golden, Colorado*, 2004.

[17] T. Knudsen, T. Bak, and M. Soltani. Distributed control of large-scale offshore wind farms. In *European Wind Energy Conference and Exhibition (EWEC) 2009*.

[18] JF Kurose and R. Simha. A microeconomic approach to optimal resource allocation indistributed computer systems. *IEEE Transactions on computers*, 38(5):705–717, 1989.

[19] S.H. Low and D.E. Lapsley. Optimization flow controlI: basic algorithm and convergence. *IEEE/ACM Transactions on Networking (TON)*, 7(6):861–874, 1999.

[20] M. Métivier. *Semimartingales: a course on stochastic processes*. Walter de Gruyter, 1982.

[21] I. Munteanu, N.A. Cutululis, A.I. Bratcu, and E. Ceanga. Optimization of variable speed wind power systems based on a LQG approach. *Control Engineering Practice*, 13(7):903–912, 2005.

[22] A. Nedic and D.P. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal of Optimization*, 12(1):109–138, 2001.

[23] A. Nedic, DP Bertsekas, and VS Borkar. Distributed asynchronous incremental subgradient methods. *Studies in Computational Mathematics*, 8:381–407, 2001.

[24] P. Novak, T. Ekelund, I. Jovik, and B. Schmidtbauer. Modeling and control of variable-speed wind-turbine drive-system dynamics. *Control Systems Magazine, IEEE*, 15(4):28–38, 1995.

[25] L.Y. Pao and K.E. Johnson. A tutorial on the dynamics and control of wind turbines and wind farms. In *American Control Conference, 2009. ACC'09.*, pages 2076–2089. IEEE, 2009.

[26] S.S. Ram, A. Nedi, and V. Veeravalli. Distributed stochastic subgradient projection algorithms for convex optimization. *Journal of Optimization Theory and Applications*, pages 1–30, 2010. 10.1007/s10957-010-9737-7.

[27] S.S. Ram, A. Nedic, and VV Veeravalli. Stochastic incremental gradient descent for estimation in sensor networks. In *Signals, Systems and Computers, 2007. ACSSC 2007. Conference Record of the Forty-First Asilomar Conference on*, pages 582–586. IEEE, 2008.

[28] S.S. Ram, A. Nedic, and VV Veeravalli. Asynchronous gossip algorithms for stochastic optimization. In *Proceedings of the 48th IEEE Conference on Decision and Control*, pages 3581–3586, 2010.

[29] A. Rantzer. Dynamic dual decomposition for distributed control. In *American Control Conference, 2009. ACC'09.*, pages 884–888. IEEE, 2009.

[30] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

[31] JG Schepers and SP Pijl. Improved modelling of wake aerodynamics and assessment of new farm control strategies. In *Journal of Physics: Conference Series*, volume 75, page 012039. IOP Publishing, 2007.

[32] M. Steinbuch, WW de Boer, OH Bosgra, S. Peters, and J. Ploeg. Optimal control of wind power plants. *Journal of Wind Engineering and Industrial Aerodynamics*, 27(1-3):237–246, 1988.

[33] J.N. Tsitsiklis and DP Bertsekas. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall Englewood Cliffs, NJ, 1989.

[34] J.N. Tsitsiklis, D.P. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–812, 1986.

[35] L. Walras. *Éléments d'économie politique pure: ou, Théorie de la richesse sociale*. R. Pichon et R. Durand-Auzias; Lausanne, F. Rouge., 1874.

[36] AD Wright and LJ Fingersh. Advanced Control Design for Wind Turbines. Technical Report NREL/TP-500-42437, National Renewable Energy Laboratory, March 2008.

[37] L. Xiao and S. Boyd. Optimal scaling of a gradient method for distributed resource allocation. *Journal of Optimization Theory and Applications*, 129(3):469–488, 2006.